

【特許請求の範囲】

【請求項1】

1つまたはそれ以上のデータ・ソースから生イベントを受け取るステップと、
前記生イベントを分類するステップと、
前記生イベントを記憶するステップと、
各生イベントに順位を付けるステップと、
2つまたはそれ以上の生イベントの間に関係を見つけ出すステップと、
2つまたはそれ以上の生イベントの間に何らかの関係を見つけ出すことに対応して、成熟
相関イベント・メッセージを生成するステップと、
生イベントの間の関係を記述する1つまたはそれ以上の成熟相関イベント・メッセージを
コンソールに表示するステップと、
を含む、セキュリティ情報を管理するための方法。

【請求項2】

各生イベントが、自動化されたシステムおよび人間の観察のうちの1つによって見つけら
れる疑わしいコンピュータの活動を含む請求項1記載の方法。

【請求項3】

前記1つまたはそれ以上のデータ・ソースから生イベントを受け取るステップが、侵入探
知システム、侵入探知システム内のディテクタ、およびファイヤ・ウォールのうちの1つ
からリアルタイムの生イベントを受け取るステップをさらに含む請求項1記載の方法。

【請求項4】

前記1つまたはそれ以上のデータ・ソースから生イベントを受け取るステップがファイル
およびデータベースの1つから生イベントを受け取るステップをさらに含む請求項1記載
の方法。

【請求項5】

前記生イベントを分類するステップが、
各生イベントのためのイベント・タイプ・パラメータを見つけ出すステップと、
前記イベント・タイプ・パラメータをリストのイベント・タイプ・カテゴリと比較するス
テップと、
各生イベントを前記リスト中の対応するイベント・タイプのカテゴリに配属するステッ
プと、
をさらに含む請求項1記載の方法。

【請求項6】

前記各生イベントに順位をつけるステップが、
各生イベントのパラメータをデータベース中の情報と比較するステップと、
各生イベントに前記生イベントの環境に追加のパラメータを付け加えるステッ
プと、
をさらに含む請求項1記載の方法。

【請求項7】

前記追加のパラメータが優先順位、脆弱度、ヒストリカル頻度値、送信元ゾーン値、宛先
ゾーン値、およびテキスト列のどれか1つを含む請求項6記載の方法。

【請求項8】

前記各生イベントに順位をつけるステップが、
生イベントの優先順位パラメータを見つけ出すステップと、
各生イベントをコンテキスト・データベースに含まれている情報と比較するステップと、
前記比較ステップに応じて一致が発生したときは各生イベントの優先順位パラメータを変
更するステップと、
前記比較ステップに応じて一致が発生しなかったときは優先順位をそのままにするステッ
プと、
をさらに含む請求項1記載の方法。

【請求項9】

10

20

30

40

50

前記2つまたはそれ以上の生イベントの間の関係を見つけ出すステップが、各生イベントを生イベントのタイプ・パラメータに対応するルールと関係づけるステップと、

1つまたはそれ以上のルールを同じ前記タイプ・パラメータを持つ生イベントのグループに適用するステップと、成功したルールの適用に基づいてコンピュータに対する攻撃またはセキュリティの侵害が起きたかどうか判断するステップと、をさらに含む請求項1記載の方法。

【請求項10】

前記生イベントを記憶するステップが、各生イベントをランダム・アクセス・メモリ（RAM）を含む高速メモリ・デバイスに記憶させるステップをさらに含む請求項1記載の方法。

【請求項11】

前記生成された前記成熟関連イベントのタイプに基づいてコンピュータに対する攻撃の意図を判断するステップをさらに含む請求項1記載の方法。

【請求項12】

メモリ管理リストを作成するステップと、各生イベントのタイムスタンプを見つけるステップと、各生イベントを前記メモリ管理リストに付け加えるステップと、をさらに含む請求項1記載の方法。

【請求項13】

1つまたはそれ以上の生イベントを監視している1つまたはそれ以上のソフトウェア・コンポーネントを見つけ出すための生イベント追跡インデックスを作成するステップをさらに含む請求項1記載の方法。

【請求項14】

第1の組のパラメータを持つ複数の生イベントを受け取るステップと、生イベント分類データベースから受け取った情報に基づいて生イベント記憶領域を設けるステップと、各イベントをイベント・タイプ・パラメータに基づいてイベント記憶領域に記憶するステップと、

各生イベントをコンテキスト・データベース中に含まれるデータと比較するステップと、コンテキスト・データベースとの前記比較に応じて各生イベントのための優先度パラメータを調整するかまたはそのままにするステップと、各生イベントを関連イベントに関係づけるステップと、前記関連イベントの連合に基づいて各イベントに1つまたはそれ以上のルールを適用するステップと、成功したルールの適用に応じて成熟関連イベント・メッセージを生成するステップと、を含む、2つまたはそれ以上のコンピュータ・イベントの間の関係を判断するための方法。

【請求項15】

各生イベントが、自動化されたシステムおよび人間の観察のうちの1つによって見つけられる疑わしいコンピュータの活動を含む請求項14記載の方法。

【請求項16】

前記コンテキスト・データベースが、脆弱度値、コンピュータ・イベントの頻度値、および送信元および宛先ゾーン値のどれか1つを含む請求項14記載の方法。

【請求項17】

前記生イベント分類データベースが、次の情報、すなわち、生イベントが1つまたはそれ以上の標的コンピュータにどのようにして影響を与える可能性があるか、生イベントによって影響を受ける可能性があるコンピュータの数はどのくらいか、および各生イベントはどのようにして1つまたはそれ以上の標的コンピュータにアクセスするか、のどれかに基

20

30

40

50

づいて生イベントをカテゴリに分ける情報を含む表を含む請求項 14 記載の方法。

【請求項 18】

複数のデータ・ソースと、

複数のデータ・ソースにリンクされたイベント・コレクタと、

前記イベント・コレクタにリンクされたフュージョン・エンジンであって、前記データ・ソースによって生成された 2 つまたはそれ以上の生イベントの間の関係を見つけたすフュージョン・エンジンと、

前記イベント・コレクタにリンクされた、前記フュージョン・エンジンによって生成された出力を表示するためのコンソールと、
を含むセキュリティ管理システム。

10

【請求項 19】

コンピュータのカーネル・モードで動作するディテクタと前記コンピュータのユーザ・モードで動作する前記フュージョン・エンジンをさらに含む請求項 18 記載のセキュリティ管理システム。

【請求項 20】

ディテクタのチップ、およびコンピュータ上で動作するソフトウェアを含む前記フュージョン・エンジンをさらに含む請求項 18 記載のセキュリティ管理システム。

【請求項 21】

ディテクタ基板、およびコンピュータ上で動作するソフトウェアを含む前記フュージョン・エンジンをさらに含む請求項 18 記載のセキュリティ管理システム。

20

【請求項 22】

コントローラと、

生イベントを受け取るためのイベント・リダと、

前記イベント・リダにリンクされた、前記受け取った生イベントを分類するためのクラシファイヤと、

前記クラシファイヤにリンクされた、生イベントの優先度を調整するためのコンテキスト・ベースのリスク調整プロセッサと、

前記コンテキスト・ベースのリスク調整プロセッサにリンクされたコンテキスト・データベースと、

2 つまたはそれ以上の生イベントの間に関係があるか否かを判断するためのルール・データベースと、

30

を含むフュージョン・エンジン。

【請求項 23】

イベント・リポータ、成熟イベントのリスト、メモリ管理リスト、および生イベント追跡インデックスをさらに含む請求項 22 記載のフュージョン・エンジン。

【請求項 24】

前記コンテキスト・データベースが、脆弱度値、コンピュータ・イベント頻度値、および送信元および宛先ゾーン値のどれか 1 つを含む請求項 22 記載のフュージョン・エンジン。

【請求項 25】

前記生イベント分類データベースが、次の情報、すなわち、生イベントが 1 つまたはそれ以上の標的コンピュータにどのようにして影響を与える可能性があるか、生イベントによって影響を受ける可能性があるコンピュータの数はどのくらいか、および各生イベントはどのようにして 1 つまたはそれ以上の標的コンピュータにアクセスするか、のどれかに基づいて生イベントをカテゴリに分ける情報を含む表を含む請求項 22 記載のフュージョン・エンジン。

40

【発明の詳細な説明】

【0001】

優先権および関連出願

本願は 2000 年 4 月 28 日に出願された米国特許出願第 60/200,316 号「ネッ 50

トワーク・セキュリティ・システムの侵入探知フュージョン・システム」というタイトルの仮特許出願の優先権を主張する。

【0002】

本願はまた2001年4月27日に出願された米国特許出願第_____号「ネットワーク上のセキュリティ・イベントを管理するシステムおよび方法」というタイトルの非仮出願（代理人整理番号05456-15005）と関連する。

【0003】

発明の属する技術分野

本発明はコンピュータ・システムおよびそのようなシステムのセキュリティに関する。より詳しく言えば、本発明は各セキュリティ・イベントを危険性に従って階級指定し、またコンピュータ・システム上または内部で発生するかもしれない2つまたはそれ以上のセキュリティ・イベントの間の関係を融合すなわち見つけ出すための方法およびシステムに関する。本発明はまた他のセキュリティに関係する情報中の関係も見つけだすことができる。

10

【0004】

従来の技術

インターネットのような分散ネットワークは、本質的に攻撃に対して脆弱である。インターネットは情報、データおよびファイルの可能な限り最も自由な交換を可能にするように設計された。しかしながら、この自由な情報の交換は代償を伴っている。多くのユーザがインターネットに接続されているネットワークおよびコンピュータを攻撃しようと試みるであろう。多くのユーザが他のユーザのプライバシーを侵害することを試みたり、また機密情報のデータベースの読取、インターネットの経路を伝わる情報の傍受を企てたりもするであろう。

20

【0005】

このようなコンピュータに対する攻撃を感知するかまたは防ぐために、情報を収集しそしてネットワーク・コンピュータのセキュリティ構成に変更を加える侵入探知システム（IDS）およびソフトウェア・プログラムが開発されている。しかしながら、これらの従来の侵入探知システムには一般的に多くの問題と難点がある。従来の侵入探知システムはたいていネットワーク上の侵入探知に専用されるハードウェアを含む。他の侵入探知システムは単にホスト・コンピュータ上で動くプログラムを含むことができる。

30

【0006】

多くの従来の侵入探知システムの問題と難点は、すべての感知設計の一部である少なくとも2つのパラメータにせいにするのであろう。第1のパラメータは、侵入探知システムのディテクタが、そのディテクタを通して流れるデータすなわち通信に対して透明であるために動作しなければならない速度である。たいてい専用のパソコン上で動作するディテクタは、ネットワークの速度が100メガビット/秒からギガビット/秒の速度にそしてそれ以上へと増しているときに、絶えず増大する大きな情報トラフィックを扱うことができなければならない。これらの高速のために、侵入探知システムのディテクタは、明白な理由によって、そのディテクタを通して流れる情報の複雑な解析を行うことができない。すなわち、もしディテクタがそれを通して流れる情報の複雑な解析を行うとしたら、そのような解析は、そのディテクタを通過する情報の流れに遅れずについていくことはできないであろう。

40

【0007】

あらゆる感知設計の一部である第2のキー・パラメータは通常ディテクタを通過するであろう情報の量である。情報がディテクタを通過する高い速度のために、ディテクタは大量のデータ・パケットを解析することができなければならない。

【0008】

現在のネットワークの速度とそのネットワーク速度の結果として発生されるそれに対応する大量の情報を考慮すると、従来の侵入探知システムの多くのディテクタは、複雑なそしてより高度化したコンピュータ攻撃に対しては非常に限られた保護しか提供できない。こ

50

の限られた保護は、侵入探知システムによって多くの誤った肯定が発生されるときに明らかになる。換言すれば、多くの従来の侵入探知システムは、なんらの脅威も攻撃も含まないコンピュータ間の通信に基づいて、偽の警報を発生する可能性がある。

【0009】

偽の警報に加えて、従来の侵入探知システムはほとんど、現在の処理速度からの限界のために、複雑な解析を行うための機能を備えていない。例えば、多くの従来の侵入探知システムは、良く知られた L o p h t C r a c k のような中央処理装置を集中的に働かせるチェックを実行することができない。L o p h t C r a c k 解読は、Windows (S M B) 接続からの暗号の試行-応答データを用いてネットワーク上で使用されているパスワードを解読することができる。L o p h t C r a c k を実行するための従来の方法は、パケット補足ツールを使ってパケットを得て、それからオフラインでパスワードを解読することである。従来の侵入探知システムはほとんどのようなリアルタイム解析においても L o p h t C r a c k の方法を用いることができなかった。

10

【0010】

従来の侵入探知システムのもう1つの障害は、ほとんどの侵入探知システムが非常に限られたまたは短期間のメモリ容量しか持たないことである。換言すれば、データ・ストリームの長いヒストリは、従来の侵入探知システムのディテクタによっては決して保持されない。

【0011】

従来の侵入探知システムのもう1つの問題は、このようなシステムのディテクタは通常単一の環境を見張るまたは観察するだけであることである。例えば、ディテクタは通常ネットワークの一部だけを観察する。従来のディテクタは、ネットワーク全体を全体として観察する代わりにネットワークの一部分だけを観察するように設計されているので、認識する範囲が限定されている。従来のディテクタはこのようにネットワークの一部分だけを監視するので、分散攻撃のようなより高度のコンピュータ攻撃を追跡することができない。

20

【0012】

より高度のコンピュータ攻撃を追跡できないことに加えて、多くの従来の侵入探知システムは、コンピュータ攻撃の攻撃者または標的の能動探索ができなかった。能動探索は通常コンピュータ攻撃がその標的に対して効果があった否かを知るための判定を行うことを含む。さらに、探索は攻撃者についての追加の情報を発見するための方法も含む。しかしながら、上述したように、多くの侵入探知システムは能動探索を許容しなかった。なぜならこのような探索はディテクタの場所を明らかにする可能性があるからである。そしてもしディテクタの場所が明らかになったら、それはしばしばコンピュータ攻撃の標的になるかもしれないからである。

30

【0013】

したがって、この分野には、ネットワーク全体のためにセキュリティ情報を管理するための方法およびシステムに対する要求がある。すなわち、この分野にはネットワーク・コンピュータ・システム内で起こり得るコンピュータ・セキュリティ・インシデンスをログに記録し、調査し、対応し、追跡することに対する要求がある。この分野にはまた、ネットワーク内またはネットワーク全体におけるセキュリティが危うくされていないか、またあるインシデントが侵入探知システムによって無視されるべきだちよって奇妙な振る舞いであるか判定することを求める要求がある。この分野には、かなり複雑なそして巧妙なコンピュータ攻撃を見つけ出され、阻止され、または防がれることができるように、複数のデータ・ソースからのセキュリティ情報を監視しそして解析することができる方法およびシステムに対する別の要求が存在する。この分野には、セキュリティ情報をリアルタイムで管理するための方法およびシステムに対するさらに別の要求が存在する。

40

【0014】

この分野には、1つまたはそれ以上のリアルタイムのコンピュータ・イベントが互いに関係があるか否か、またそれらがより大きな企てまたは巧妙な攻撃の一部であるか否かを判定することができるようなセキュリティ情報を管理するための方法およびシステムに対す

50

るさらに別の要求がある。この分野には、複数のコンピュータ・イベントが、それらがより大きな企てすなわち攻撃の一部であるならば、互いに相関づけられることができるセキュリティ情報を管理するための方法およびシステムに対する追加の要求がある。この分野には、検出されるコンピュータ・イベントに、ネットワークまたは個々のコンピュータに対して最も大きな損害を引き起こすかもしれないコンピュータ・イベントに注意を集中させることができるように優先順位を付けることができるセキュリティ情報を管理するための方法およびシステムに対するもう1つの要求がある。同様に、この分野には、単一のコンピュータ攻撃から派生するか、またはそれから発生させられるかもしれない追加のコンピュータ攻撃の防止に加えて、既存のコンピュータ攻撃に対する迅速な対応を可能にするセキュリティ情報を管理するための方法およびシステムに対するさらに1つの要求がある。この分野には、リアルタイムのコンピュータ・イベントが、そのイベントが発生した環境のコンテキストの中で、それらの優先度に従って分類されかつ階級が決められるようなセキュリティ情報を管理するための方法およびシステムに対するさらに別の要求がある。

【0015】

発明の概要

本発明は、ネットワーク接続されたコンピュータ・システム内で発生し得るコンピュータのセキュリティに関係するインシデントを記録し、調査し、対応し、そして追跡することができるコンピュータ・セキュリティ管理システムを提供することにより、上記の問題を解決することができる。本発明は、疑わしいコンピュータの活動または実際のコンピュータのセキュリティに対する脅威を追跡することができる。実際のコンピュータのセキュリティに対する脅威には、コンピュータまたはコンピュータ・ネットワークに対する完全性攻撃、秘密性攻撃、サービス拒否攻撃、多段攻撃、または他の類似の攻撃が含まれるが、これに限定されない。本発明は、典型的には、データ・ソースから得られる疑わしいコンピュータ活動の記述をリアルタイム生イベントと呼び、実際のコンピュータ・セキュリティにたいする脅威を成熟相関イベントと呼ぶ。本発明は、1つまたはそれ以上のデータ・ソースから収集されたセキュリティ情報を管理するための方法とシステムを含むことができる。より具体的に言えば、本発明は、データ・ソースによって行われる処理の速度を低下させることなく、悪意のある振る舞いを示すかも知れない生イベントの間の関係を検出して1つまたはそれ以上のコンソールに整理された情報の提示を行うために、複数のデータ・ソースからの情報を「融合」すなわち集めて整理しそしてこの情報を解析するフュージョン・エンジンを含む。

【0016】

複数のデータ・ソースはネットワークのトラフィックまたは個々のコンピュータまたはその両方を監視するセンサまたはディテクタを含むことができる。センサは侵入探知システム（IDS）と呼ばれることもできるデバイスを含むことができる。本発明はIDSデバイスとは別であるので、大量のデータのやりとりをリアルタイムで処理することが重要となるときにIDSデバイスが効率的にかつ高速で動作することを可能にする。

【0017】

データ・ソースはまたファイヤ・ウォールおよび他の類似のセキュリティまたはIDSデバイスも含むことができる。さらに、データ・ソースは、関心を持つネットワークまたはコンピュータについての追加の環境情報を提供する追跡記録システムのような、リアルタイム情報を提供することができるまたはできないどんなデバイスを含んでもよい。例えば、1つのデータ・ソースはデータベースを含むことができる。データベースは、タイプの異なる生イベントのカテゴリを含む、生イベント分類データベースを含んでもよい。他のデータベースは、ホストの脆弱状態、ヒストリカルなコンピュータ・イベントの頻度値、およびネットワークのゾーンの設定のようなネットワークのコンテキスト（状況）を示す情報を含むコンテキストまたは知識データベースを含むことができる。

【0018】

複数のデータ・ソースから、本発明のフュージョン・エンジンは、リアルタイムの生のコ

10

20

30

40

50

ンピュータ・イベントを互いに関係づけ、分類することができる。すなわち、通常かなりの時間の後コンピュータ・イベントを処理する従来の技術と異なり、本発明は、1つまたはそれ以上のリアルタイムの生のコンピュータ・イベント間の関係を、それらをリアルタイムで受け取りながら、見つけ出すことができる。リアルタイムの生のコンピュータ・イベントすなわち生イベントは、1台のコンピュータまたは複数のコンピュータに対する攻撃である可能性があるものとして侵入探知システムによって追跡されるかも知れないすべてのコンピュータの活動を含むことができる。生イベントは侵入探知システムのディテクタによって生成されることができる。各生イベントは、コンピュータの活動の送信元インターネット・プロトコル・アドレス、コンピュータの活動の宛先インターネット・プロトコル・アドレス、ディテクタによって指定された優先順位、ディテクタによって指定された脆弱度、タイムスタンプおよびイベント・タイプ・パラメータなどを含む。ただしそれらに限定されず、いろいろなパラメータを持つことができる。

【0019】

フュージョン・エンジンは、1つまたはそれ以上のリアルタイムの生イベントが互いに関係しているか否か、およびそれらがより大きなくらのコンピュータ攻撃の一部かそうでないか判断することができる。互いに関係があり、コンピュータ攻撃が起こりつつあるかもしれないことを示しているかもしれないリアルタイムの生イベントは、フュージョン・エンジンによって、成熟関連イベントと呼ばれる。1つの関連イベントは、1つまたはそれ以上の生イベントを含むことができる。しかしながら、関連イベントは、実際のセキュリティに対する脅威または攻撃が探知されたことを意味しない。関連イベントは典型的には関係のある生イベントを記憶し、通常その関連イベントが成熟したと見なされるときにセキュリティ・イベントすなわちコンピュータ攻撃が起こったことを示す。成熟しているとは見なされるためには、関連イベントは対応する関連ルールの判断基準またはアルゴリズムを満足しなければならない。したがって、成熟関連イベントまたは実際のコンピュータ・セキュリティに対する脅威またはコンピュータ攻撃として認められていない、1つまたはそれ以上の生イベントを含む多数の関連イベントを追跡することが可能である。

【0020】

フュージョン・エンジンは、イベントが発生した環境またはコンテキストについての情報に基づいて、成熟関連イベントと同じくリアルタイム生イベントのリスクを評価し順位づけることもできる。フュージョン・エンジンはこの危険および順位情報をメッセージとしてコンソール上に表示することができる。フュージョン・エンジンは、成熟関連イベントの更新を生成し、コンソールに送ることができる。さらに、フュージョン・エンジンは、成熟関連イベントが発生しなくなったときに、それを検出して表示する。

【0021】

リアルタイム生イベントのリスクを評価し順位づけるために、フュージョン・エンジンは、前述した生イベント分類データベースおよび知識データベースを利用することができる。生イベント分類データベースはフュージョン・エンジンが生コンピュータ・イベントを分類することを可能にし、知識データベースはフュージョン・エンジンがその生コンピュータ・イベントのコンテキストに基づいて生コンピュータ・イベントの危険性を順位づけ評価することを可能にする。生イベント分類データベースは1つまたはそれ以上のセキュリティ情報の表を含むことができる。すなわち、生イベント分類データベースは、生イベントを、それらが標的のホストに与える衝撃（秘密性、完全性、または利用可能性）、範囲（ネットワーク、ホスト、またはサービス）、およびそれらが使う方法（裏口からの侵入、IDSの回避すなわち探知の回避、その他）に基づいて分類することができる情報を含む表を備えることができる。生コンピュータ・イベントのコンテキストは、その生イベントのパラメータを、コンテキストすなわち知識データベース中のコンテキスト・パラメータ、例えば前述したイベントの脆弱度、ヒストリカル・コンピュータ頻度値、およびゾーン設定など、と比較することにより、決定できる。

【0022】

1つまたはそれ以上の生コンピュータ・イベントが成熟コンピュータ・イベントの一部で

10

20

30

40

50

あるか、またはそれを形成しているか判断するために、フュージョン・エンジンは、フュージョン・エンジンが生コンピュータ・イベントを分類するやり方に基づいて始動される1つまたはそれ以上のルールを適用することができる。換言すれば、フュージョン・エンジンによって適用されるルールは、生イベントの分類(タイプまたは種類の識別)にしたがって活性化され、生イベントに適用される。

【0023】

生コンピュータ・イベントが1つの成熟コンピュータ・イベントの一部であるかまたはそれを形成しているか、または本当にセキュリティの脅威かを判断することに加えて、フュージョン・エンジンはその高速メモリ資源を非常に効率的に管理する。例えば、フュージョン・エンジンは、予め決められた時間を越えている、または予め決められた条件に適合している、またはその両方の生イベント、未成熟および成熟関連イベントを消去するメモリ管理技術を採用することができる。この高速メモリ資源は、生イベントおよび成熟関連イベントの分類に従って分けられているデータを記憶するRAMを含むことができる。

10

【0024】

実施の形態の詳細な説明

本発明は分散コンピューティング環境内で動くプログラム・モジュールとして実現することができる。本発明は、コンピュータのセキュリティ事件を記録し、調査し、対応し、追跡することができるコンピュータ・セキュリティ管理システムを含むことができる。本発明は、複数のコンソールに対してまとめられた、そして時にはランク付けされた情報の提示を行うために、複数のデータ・ソースからの情報を「融合(フュージョン)」すなわち組み立てるフュージョン・エンジンを含むことができる。このフュージョン・エンジンは、1つまたはそれ以上のデータベースに基づいてリアルタイムのコンピュータ・イベントをランクづけしながら、未加工のリアルタイムのコンピュータ・イベントを分類することができる。

20

【0025】

説明のための動作環境

説明のための動作環境はパーソナル・コンピュータおよびサーバ上で動作するプログラム・モジュールの面において一般的に説明されるであろうが、この分野の専門家は、他のオペレーティング・システム・プログラムとの組合せまたは他のタイプのコンピュータのための他のタイプのプログラム・モジュールとの組合せにおいて実現できること分かるであろう。さらに、この分野の専門家は、他のオペレーティング・システム・プログラムとの組合せまたは他のタイプのコンピュータのための他のタイプのプログラム・モジュールとの組合せにおいて実現できることを理解するであろう。さらに、この分野の専門家は、スタンドアロン環境または分散コンピューティング環境またはその両方において実現できることを理解するであろう。分散コンピューティング環境では、プログラム・モジュールは、物理的にいろいろなローカルおよびリモートの記憶装置に置かれてもよい。プログラム・モジュールの実行は、スタンドアロン方式でローカルに、またはクライアントサーバ方式でリモートに行われてもよい。このような分散コンピューティング環境の例には、ローカル・エリア・ネットワークおよびインターネットがある。

30

【0026】

この後の詳細な説明は、大部分が、処理ユニット(プロセッサ)、記憶装置、接続された表示装置および入力装置を含む、従来のコンピュータの構成要素による処理および動作の記号表示によりなされる。さらに、これらの処理および動作は、リモート・ファイル・サーバ、コンピュータ・サーバおよび記憶装置を含む、異種分散コンピューティング環境における従来のコンピュータ構成要素を利用してよい。これらの従来の分散コンピューティングの構成要素の各々は、通信ネットワークを通してプロセッサによりアクセスできる。

40

【0027】

コンピュータにより行われる処理および動作は、プロセッサによる信号の操作および1つまたはそれ以上の記憶装置に常駐するデータ構造内にこれらの信号の保持を含む。この説

50

明のために、処理は、一般的に、望みの結果をもたらす一連のコンピュータにより実行されるステップであるとされている。これらのステップは通常物理量の物理的な操作を必要とする。通常は、必ずそうであるわけではないが、これらの量は、記憶され、転送され、組み合わせられ、比較され、または他の方法で操作されることができる電気、磁気、または光の信号の形を取る。この分野の専門家は、慣習で、これらの信号の表現をビット、バイト、ワード、情報、要素、記号、文字、数、点、データ、項目、オブジェクト、イメージ、ファイルなどと呼んでいる。しかしながら、これらのおよび類似の語は、コンピュータの動作のための物理量に関係づけられていること、およびこれらの語はコンピュータの内部および動作中に存在する物理量に付けられた約束ごとのラベルにすぎないことは憶えておかれるべきである。

【0028】

またコンピュータの内部における操作は、しばしば人間の操作者によって行われる手操作と関係づけられる、作成る、加える、計算する、比較する、移動させる、受け取る、判断する、識別する、移植する、ロードする、実行するなどのような言葉でしばしば言及されることも理解されるべきである。ここで説明されている操作は、人間の操作者すなわちコンピュータと対話するユーザによって与えられたいろいろな入力に関係して行われるマシンの動作でもよい。

【0029】

さらに、ここに説明されているプログラム、プロセス、メソッドなどは、特定のコンピュータまたは装置に関係づけられても、限定されてもいないことは理解されるべきである。むしろ、いろいろなタイプの汎用マシンが、ここに説明されている教示に従って作られたプログラム・モジュールと共に使用できる。同様に、配線論理またはリード・オンリ・メモリのような不揮発メモリに記憶されたプログラムを持った、特定のネットワーク・アーキテクチャの中の専用のコンピュータにより、ここに記載されている方法のステップを実行するための特殊化された装置を作ること、有利であることがわかるであろう。

【0030】

さて図面を参照して、本発明のいろいろな面と説明のための動作環境を説明する。なお幾つかの図を通じて類似の数字は類似の要素を示す。

【0031】

図1および下記の説明は、本発明を実施するのに適したコンピュータ環境の短い一般的な説明を与えることを意図している。図1を参照すると、本発明を実施するための説明のための環境は、処理ユーザ102、リード・オンリ・メモリ（ROM）104およびランダム・アクセス・メモリ（RAM）108を含むシステム・メモリ、システム・メモリを処理ユニット102に結合するシステム・バス105を含む従来のパーソナル・コンピュータ100を含む。リード・オンリ・メモリ（ROM）104は、例えば起動中にパーソナル・コンピュータ100の内部の要素間のデータの転送を助ける基本ルーチンを内蔵する、基本入出力システム106（BIOS）を含む。パーソナル・コンピュータ100はさらにハード・ディスク・ドライブ118と、例えばCD-ROMディスクまたはDVDディスクを読むため、または他の光メディアに対する読出しまたは書き込みのための光ディスク・ドライブ122を含む。これらのドライブおよびそれらで使用されるコンピュータが読取り可能なメディアは、パーソナル・コンピュータ100のための不揮発記憶を提供する。上のコンピュータが読取り可能なメディアの種類はハード・ディスク、取外し可能な磁気ディスクおよびCD-ROMまたはDVD-ROMディスクのことを言っているが、磁気カセット、フラッシュ・メモリ・カード、デジタル・ビデオ・ディスク、ベルヌイ・カートリッジなどのコンピュータにより読みとり可能な他のタイプのメディアもこの説明のための動作環境において使用できることは、この分野の専門家にはわかるはずである。

【0032】

オペレーティング・システム114およびWWWブラウザ112のようなワールド・ワイド・ウェブをブラウズするためのプログラムなどの1つまたはそれ以上のアプリケーション

10

20

30

40

50

ン・プログラム110を含む複数のプログラム・モジュールが、ドライブやRAM108に記憶されていてもよい。このようなプログラム・モジュールは、ハード・ディスク・ドライブ118に記憶されていて、実行のためにその一部または全部がRAM108にロードされてもよい。

【0033】

ユーザは、キーボード128およびマウス130のようなポインティング・デバイスで、コマンドや情報を入力できる。多の制御入力装置（図示されていない）には、マイクロホン、ジョイスティック、ゲーム・パッド、衛星アンテナ、スキャナなどが含まれる。これらや他の入力装置はしばしばシステム・バスに接続された入力／出力インタフェース120を通じて処理ユニット102に接続されるが、ゲーム・ポート、ユニバーサル・シリアル・バス、またはファイヤ・ポートなどの他のインタフェースによって接続することもできる。ディスプレイ・モニタ126または他のタイプのディスプレイ装置も、ビデオ・ディスプレイ・アダプタ116のようなインタフェースを介してシステム・バス105に接続される。モニタに加えて、パーソナル・コンピュータは通常、スピーカやプリンタのような他の周辺出力装置（図示されていない）を含む。パーソナル・コンピュータ100はモニタ126にグラフィカル・ユーザ・インタフェースを表示することができてよい。

【0034】

パーソナル・コンピュータ100は、1つまたはそれ以上の、ホスト・コンピュータ140のようなリモート・コンピュータに対する論理接続を使って、ネットワーク化された環境で動作してもよい。ホスト・コンピュータ140はサーバ、ルータ、ピア・デバイスまたは他の通常のネットワーク・ノードであればよく、そして通常パーソナル・コンピュータ100に関係するとされた要素の多くまたは全てを含む。LAN136はさらにインターネット138へのアクセスのためにインターネット・サービス・プロバイダ（ISP）に接続されていてもよい。このように、WWWブラウザ112は、LAN136、ISP134およびインターネット138を通してホスト・コンピュータ140に接続することができる。このようなワートワーキング環境は、オフィス、企業全体にわたるコンピュータ・ネットワーク、イントラネットおよびインターネットにおいてありふれている。

【0035】

LANネットワーク環境で使用されるとき、パーソナル・コンピュータ100はネットワーク・インタフェース・ユニット124を介してLAN136に接続される。WANネットワーク環境で使用されるとき、パーソナル・コンピュータ100は、通常、インターネット・サービス・プロバイダ134を通してインターネットに対する通信を確立するためのモデム132または他の手段を含む。モデム132は、内蔵または外置きのどちらでもよく、入力／出力インタフェース120を介してシステム・バス105に接続される。図示されているネットワーク接続は説明のためのものであり、コンピュータ間の通信リンクを確立するための他の手段も使用できることはわかるであろう。

【0036】

オペレーティング・システム114は入力／出力動作を含む上述したパーソナル・コンピュータ100の動作を全体的に制御する。この説明のための動作環境では、本発明はマイクロソフト社の「Windows NT」オペレーティング・システムおよびWWWブラウザ112と共に使われている。しかしながら、本発明は、マイクロソフト社の「Windows 3.1」、「Windows 95」、「Windows 98」および「Windows 2000」オペレーティング・システム、IBM社の「OS/2」および「AIX」オペレーティング・システム、サン・マイクロシステムズにより製造されるワークステーションに使用されるサンソフト社の「SOLARIS」オペレーティング・システム、アップルコンピュータ社により製造される「MACHINTOSH」コンピュータに使用されるオペレーティング・システムなどの他のオペレーティング・システムにおいて使用するために実現することもできる。同様に、本発明は、この分野の専門家に知られている他のWWWブラウザと共に使用するために実現することもできる。

10

20

30

40

50

【0037】

ホスト・コンピュータ140もインターネット138に接続され、上述したパーソナル・コンピュータ100に含まれるものと類似の構成要素を含んでもよい。さらに、ホスト・コンピュータ140は、WWWページを求める要求を受け取ったり、WWWサーバ142のような要求者に対してそのようなページを送ったりするアプリケーション・プログラムを実行してもよい。WWWサーバ142は、WWWブラウザ112からのWWWページ150または他の文書を求める要求を受け取ることができる。これらの要求に応じて、WWWサーバ142は、ハイパーテキスト・マークアップ・ラングウェッジ(「HTML」)またはイー・エクステンシブル・マークアップ・ラングウェッジ(XML)のような他のマークアップ・ラングウェッジのファイルを含むWWWページ190をWWWブラウザ112に送信することができる。同様に、WWWサーバ142は、グラフィック・イメージまたはテキスト情報などの要求されたデータ・ファイル148をWWWブラウザ112に送信することもできる。WWWサーバ142はまたWWWブラウザ112に対して送信するためのWWWページ150を動的に生成するために、CGI、PERL、ASP、またはJSP(ジャバ・サーバ・ページ)スクリプトなどのスクリプト144を実行することもできる。WWWサーバ142はまた、ジャバスクリプトで書かれたスクリプトのようなスクリプト144を、実行のために、WWWブラウザ122に送信することもできる。

【0038】

同様に、WWWサーバ142はサン・マイクロシステムズ社によって開発されたジャバ・プログラミング言語で書かれたプログラムを、実行のためにWWWブラウザ112に送信することもできる。WWWサーバ142はパッッシュまたはネットスケープ・ウェブサーバを走らせるUNIXプラットフォームを含むことができる。それは別に、WWWサーバ142はインターネット・インフォメーション・サーバ(IIS)を含むことができる。本発明はこれらの列挙された例に限定されない。他のウェブ・サーバ環境も本発明の範囲を超えていない。

【0039】

以下により詳しく説明するように、本発明のいくつかの面は、スクリプト144のような、ホスト・コンピュータ142によって実行されるアプリケーション・プログラムで実現されることもできるし、ジャバ・アプリケーション146のような、コンピュータ100によって実行されるアプリケーション・プログラムで実現されることもできる。この分野の専門家は、本発明のいくつかの面がスタンドアロンのアプリケーションで実現されることもできることを理解するであろう。

【0040】

代表的なコンピュータ・アーキテクチャ

図2を参照して、本発明の1つの代表的な実施の形態のためのコンピュータ・アーキテクチャを説明する。図2は、1つまたはそれ以上のデータ・ソースから集められた安全情報を管理するためのシステム20を示している。セキュリティ・システム20は、イベント・コレクタ24にリンクされたフュージョン・エンジン22を含むことができる。イベント・コレクタ24は、イベント・シンクすなわち複数のデータ・ソースから受け取ったイベントを論理的なやり方で整理することができる装置を含むことができる。イベント・コレクタ24のさらに詳しいことは、参照によりここに組み入れられている2001年4月27日に提出された米国特許出願第_____号「ネットワーク上でのセキュリティ・イベントを管理するシステムおよび方法」というタイトルの関係する出願(代理人整理番号05456-15005)に説明されている。

【0041】

セキュリティ管理システム20はさらに、同じくイベント・コレクタ24にリンクされているイベント・データベース26を含むことができる。セキュリティ管理システム20はまた、イベント・コレクタ24にリンクされたデータ・ソース28および同じくイベント・コレクタ24にリンクされたコンソール30を含むことができる。データベースからの情報は、死どの場合、ランダム・アクセス・メモリ(RAM)のような高速のメモリ・デ

10

20

30

40

50

バイスを含むフュージョン・エンジン２２にロードされる。未加工データとデータベースの比較は非常に迅速にかつ非常に効率的に行われなければならないからである。フュージョン・エンジン２２において使われる大部分のメモリ資源はＲＡＭ（これ以後「キャッシュ」と呼ばれることもある）のような高速のメモリ・デバイスを含む。しかしながら、他のメモリ資源は本発明の範囲に含まれている。フュージョン・エンジン２２のメモリ資源は、大量の情報をより高速で処理するように設計されなければならない。

【００４２】

１つまたはそれ以上のデータ・ソース２８は、多くのいろいろなハードウェアおよびソフトウェア・デバイスを含むことができる。例えば、データ・ソース２８はネットワーク・ディテクタまたはホスト・ディテクタを含むことができる。同様に、データ・ソース２８はまたファイヤ・ウォールまたは追跡記録システムも含むことができる。本発明は、図に示したタイプのデータ・ソースに限定されない。データ・ソース２８の機能は、イベント・コレクタ２４にいろいろなタイプの情報を提供することである。その情報がセキュリティ管理システム２０により監視されているネットワーク、ホスト、または単一のコンピュータに関係しているかも知れないからである。他の類似のデータ・ソース２８も本発明の範囲に含まれる。１つのデータ・ソース２８は、データ・パケットの形のネットワーク・トラフィックを監視するホスト・ディテクタを含むことができる。もう１つのデータ・ソース２８は、ネットワークまたはコンピュータの活動を監視しているユーザによって行われる観察を含むことができる。

【００４３】

１つまたはそれ以上のデータ・ソース２８は、それらの情報をイベント・コレクタ２４に送る。イベント・コレクタ２４は、１つまたはそれ以上のデータ・ソース２８から受け取ったデータを記憶しそして収集するように設計された１つまたはそれ以上のプログラム・モジュールを含んでもよい。イベント・コレクタ２４はそのデータを整理してイベント・データベース２６に記憶する。イベント・コレクタ２４はまたデータ・ソース２８から受け取った情報を何でもフュージョン・エンジン２２に転送する。侵入探知システムのディテクタ２８は、予め決められたパターンを探して未加工のネットワーク・トラフィックまたはローカル・システム・イベントを細かく調べる。ディテクタが情報をこれらの予め決められたパターンと同じであるとみなしたら、ディテクタは未加工のイベントを発生する。このイベントは次にイベント・コレクタに送られ、さらにその後フュージョン・エンジン２２に送られる。フュージョン・エンジンはイベント・コレクタ２４から受け取った未加工のイベントすなわち情報を組み立てるすなわち「融合（フュージョン）」する。換言すると、フュージョン・エンジン２２は、互いに関係がある未加工のコンピュータ・イベントの相関関係を調べる（それらの間の関係を明らかにする）ことにより整理された情報の提示を行うために、１つまたはそれ以上のデータ・ソース２８から受け取った情報を整理し解析する。

【００４４】

フュージョン・エンジン２２が２つまたはそれ以上のイベントが互いに関係がある（「相関」イベントを形成している）と判断したら、フュージョン・エンジン２２はメッセージを生成し、これらのメッセージをイベント・コレクタ２４に送る。イベント・コレクタ２４は今度はそのフュージョン・エンジン２２により生成されたメッセージをコンソール３０に送る。

【００４５】

コンソール３０は単独のパーソナル・コンピュータ上で動くプログラム・モジュールを含んでもよい。フュージョン・エンジン２２はパーソナル・コンピュータ上で動く１つまたはそれ以上のプログラム・モジュールを含んでもよい。フュージョン・エンジン２２、イベント・コレクタ２４、およびイベント・データベース２６は、これらのソフトウェア・コンポーネントの各々が１つのコンピュータ上にあることを表すために、四角形で囲まれている。しかしながら、本発明はこの構成に限定されない。そしてそれゆえに、フュージョン・エンジン２２、イベント・コレクタ２４、およびイベント・データベース２６は、

10

20

30

40

50

別々のコンピュータ装置に在ってもよい。図示されているソフトウェア・コンポーネントの他の組合せも実現できる。逆に言えば、イベント・コレクタ24とイベント・データベース26は1つのハードウェア・デバイスは1つのハードウェア装置上に存在することができるが、フュージョン・エンジン22は別のハードウェア装置上に存在する。この分野の専門家は、開示されるソフトウェアのアーキテクチャは図面に示されているアーキテクチャに限定されないことを理解するであろう。

【0046】

次に図3を参照すると、本発明の別の代表的なソフトウェア・アーキテクチャを示す機能ブロック図が示されている。図3では、フュージョン・エンジン22のプログラム・モジュール22とディテクタ・モジュール28のようなデータ・ソースが、単一のマシンに在ることができる。すなわちディテクタ28の高速のIDS機能は、コンピュータのカーネルの近くに常駐することができるが、フュージョン・エンジン22はコンピュータのユーザ・モード部に常駐することができる。このように、さらに加わったフュージョン・エンジン22の処理は、ディテクタ24によって行われる高速の侵入検知システムの機能の速度を低下させないであろう。

【0047】

次に図4を参照して、この図は、本発明のための代表的なソフトウェアおよびハードウェア・アーキテクチャのもう1つの機能ブロック線図を示す。この1つの例としての実施の形態においては、ディテクタを含むデータ・ソース28は、高速侵入検知システムの機能が果たされ得るように、検出基板または検出チップのようなハードウェア・デバイスの形で実現され得るであろう。この説明のための実施の形態では、フュージョン・エンジン22は単にソフトウェア中にプログラム・モジュールとして常駐することができるであろう。図4は、高速データ・ストリームに対するアクセスを必要とするデータ・ソース28は、ネットワークの処理速度が、著しいインタープリテーションまたは遅延またはその両方無しに、達成され得るように、フュージョン・エンジン22から切り離され得ることを示している。

【0048】

次に図5Aを参照して、この図は、コンピュータのインシデンス・ソース500についての情報をやはりフュージョン・エンジン22に接続されているイベント・コレクタ24に供給するセキュリティ情報のデータ・ソース28の機能ブロック図を示す。図5Aはさらに、多数のデータ・ソース28、ユーザ・ワークステーション520、コンピュータ・インシデンス・ソース500の対象であるサーバ530、内部ルータ540、およびサーバ550を含んでもよいネットワーク510を示している。ネットワーク510は外部ルータ580およびファイヤ・ウォール28によってインターネット590に接続されている。ファイヤ・ウォール28はパシジョン・ホストまたは類似の装置を含むことができる。ファイヤ・ウォール28はまた、その内部スクリーニング・ルータ540に入出する全てのパケットのデータを調べられるかもしれない内部スクリーニング・ルータ540に接続されることもできる。ユーザ・ワークステーション520は、サーバ530、550にアクセスするスタンドアロンのパーソナル・コンピュータであってもよい。

【0049】

コンピュータ・インシデント・ソース500は、ネットワーク510およびより具体的にはサーバ530（攻撃されたホスト）に対する攻撃を発するコンピュータまたはコンピュータのネットワークであってもよい。コンピュータ・インシデント・ソース500はローカル・エリア・ネットワークのサーバ560に接続されてもよい。または、サーバ560の代わりに、コンピュータ・インシデント・ソース500は、ダイヤルアップ・インターネット・サーバ・プロバイダ（ISP）またはインターネットに接続された他のコンピュータに接続されることもできる。サーバ560またはISP（またはインターネットに接続された他のコンピュータ）は次にルータ570に接続され得る。ルータ570はインターネット590のような分散コンピュータ・ネットワークに対するアクセスを提供する。

【0050】

コンピュータ・インシデント・ソース500はネットワーク510の外にあってもよいが、ネットワーク510の内部にあることも可能である。すなわち、コンピュータ・インシデント・ソース500はネットワーク510の内部にあるユーザ・ワークステーション520であってもよい。例えば、社内に不満をいだく社員がいる場合には、ユーザ・ワークステーション520は、その社員がネットワーク510または1つまたはそれ以上の他のワークステーション520の動作を妨害する決心をするとき、コンピュータ・インシデント・ソース500として使われ得る。

【0051】

データ・ソース28の各々は、破線により示される、イベント・コレクタ24に入るデータ・ラインを持つ。しかしながら、この破線のデータは実際の物理ラインを含むこともできる。しかしながら、これらのデータ・ラインは、データ・ソースの各々が操作によりイベント・コレクタ24が操作によりイベント・コレクタ24にリンクされることを示す目的のためである。また、イベント・コレクタ24は、コンピュータ・インシデント・ソース500からの直接の攻撃に対して脆弱でないように、ネットワーク510内に存在することができる。図5内でのイベント・コレクタ24の配置は、イベント・コレクタ24の収集機能を説明する。図5は、このようなシステムを支援するために実現されるであろう実際の物理アーキテクチャというよりむしろセキュリティ情報を管理するためのシステムの基本的な概念を示している。

【0052】

フュージョン・エンジンによって処理されるデータの説明例

次に図5Bを参照する。この図は、侵入探知システムのディテクタにより発生される代表的な生イベント505である。生イベント505は、送信元のインターネット・プロトコル・アドレス515、宛先のインターネット・プロトコル・アドレス515、優先度535、ディテクタにより指定された脆弱性545、イベント・タイプ・パラメータ555、およびタイムスタンプ565を含む。以下にさらに詳しく説明されるであろうように、侵入探知システムのディテクタによって指定される優先度535は、本質的に通常は非常に控えめである。すなわち、ディテクタは情報を非常に素早く処理しなければならないので、複雑なアルゴリズムを走らせることも、あるコンピュータの生イベントの危険性を確認するためのテストを行うこともできない。したがって、ディテクタにより発生される多くの生イベントの優先度55は、生イベントの実際の優先度と比べて非常に高いであろう。

【0053】

次に図5Cを参照する。この図はCoBRA (Context Based Risk Adjustment: コンテキスト・ベースの危険度の調整) により処理された生イベントを示す説明図である。CoBRAにより処理された生イベント502は通常は先にディテクタにより指定された生イベントのパラメータの全てと、さらにCoBRAにより指定されるパラメータを含む。CoBRAにより指定されたパラメータは、次のどれでも含むことができる。すなわち、CoBRAにより指定された脆弱性値504、CoBRAにより指定されたヒストリカル頻度506、CoBRAにより指定されたソース・ゾーン値508、CoBRAにより指定されたデスティネーション・ゾーン値510、CoBRAにより指定されたセンサ・ゾーン値512、CoBRAにより指定されたオリジナル優先度514、および生イベントの優先度が調整された理由を含む優先度変更理由516デキストリ (調整された場合)。これらのCoBRAにより指定される値は、この後で図11および図12に関してさらに詳しく説明されであろう。

【0054】

フュージョン・エンジンにより処理される生および相関イベントの例

次に図5Dを参照する。この図は代表例的な、攻撃されたホストからの攻撃 (Attack From Attacked Host: AFAH) コンピュータ・セキュリティの脅威を示す機能ブロック図である。図5は、インターネット・プロトコル・アドレス2.2.2.2を持つホスト (攻撃されたホスト) に攻撃を送るインターネット・プロコ

10

20

30

40

50

トル・アドレス 1. 1. 1. 1. を持つコンピュータ・インシデント・ソース 503 を示す。コンピュータ・インシデント・ソース 503 と攻撃を受けたホストの間の攻撃は、生コンピュータ・イベント I として特徴づけられることができる。攻撃を受けた後、攻撃を受けたホスト 505 は次にインターネット・プロトコル・アドレス 3. 3. 3. を持つ第 2 のホスト 507 に別の攻撃を送る。コンピュータ・インシデント・ソース 503 と攻撃を受けたホストの間の攻撃は、生コンピュータ・イベント I として特徴づけられるであろう。攻撃を受けた後、攻撃を受けたホスト 505 は次にインターネット・プロトコル・アドレス 3. 3. 3. を持つ第 2 のホスト 507 に別の攻撃を送る。攻撃を受けたホスト 505 と第 2 のホスト 507 の間の攻撃は、第 2 の生コンピュータ・イベント I I として特徴づけられるであろう。第 2 のホスト 507 はインターネット・プロトコル・アドレス 4. 4. 4. 4. を持つ第 3 のホスト 509 に対する攻撃を発生する。第 2 のホスト 507 と第 3 のホスト 509 の間の攻撃は、第 3 の生コンピュータ・イベント I I I として特徴づけられるであろう。

【0055】

生イベント I, I I, I I I を処理した後、フュージョン・エンジン 22 はそれぞれの生イベントの間の関係を見つけるであろう。したがって、図 3 に示される生イベントを処理した後、フュージョン・エンジンは、第 1 と第 2 の生イベント I および I I に対応する成熟相関イベント 511 を発生するかも知れない。フュージョン・エンジン 22 は、第 2 と第 3 の生イベント I I および I I I の間の関係を特定する第 2 の成熟相関イベント 511 を発生するかも知れない。第 1 および第 2 の成熟相関イベント 511 および 513 を発生するためにフュージョン・エンジン 22 によって行われる処理のさらに詳しい詳細は、以下に図 7 および図 14 に関してさらに詳しく説明されるであろう。

【0056】

次に図 5 E を参照する。この図は、図 15 にもとづく 1 例の相関イベントのあり得るデータを示す説明図である。図 5 E に示される相関イベント 511 は、2 組のリストを含むであろう。第 1 のリストは、攻撃を受けたホスト 505 に関係する入ってくる攻撃および攻撃を受けたホスト 505 に関係する出ていく攻撃を示すかも知れない。第 1 の例の相関イベント 511 のより詳しいことは、図 7 および図 14 に関して以下にさらに詳しく説明されるであろう。

【0057】

次に図 5 F を参照する。この図は、図 15 に示されている第 2 の例の相関イベント 513 の考えられ得るデータを示す説明図である。第 2 の相関イベント 513 も 2 つのリスト、すなわち第 2 のホスト 507 に関係する入ってくる攻撃を示す第 1 のリストと第 2 のホスト 507 に関係する出ていく攻撃を示す第 2 のリスト、から成る。第 2 の例の相関イベント 511 のより詳しいことは、図 7 および図 14 に関して以下にさらに詳しく説明されるであろう。

【0058】

図 5 D から 5 F により説明されている攻撃を受けたホストからの攻撃というコンピュータ・セキュリティに対する脅威の例は、フュージョン・エンジン 22 で解析できる考え得るコンピュータ・セキュリティに対する脅威の 1 例にすぎない。上および以下に説明されるように、他のタイプのコンピュータ・セキュリティに対する脅威も本発明の範囲を逸脱しない。1 つの例としての実施の形態では、フュージョン・エンジンは少なくとも 20 の異なるタイプの可能性のある相関イベントを追跡するかも知れない。この分野に熟練した人々は、本発明が図 5 D に示される説明のための相関イベントに限定されないこと、そしてより少ないまたはより多い相関イベントが、本発明の範囲と趣旨から逸脱することなく、本発明によって利用され得ることを理解するであろう。

【0059】

説明のためのフュージョン・エンジンのソフトウェア・コンポーネント

図 6 は、図 2 に示されているフュージョン・エンジン 22 の幾つかの構成要素を示す機能ブロック図である。基本的には、図 6 はフュージョン・エンジン 22 のアーキテクチャを

10

20

30

40

50

作り上げている複数のソフトウェア・コンポーネントのうちの幾つかを示している。

【0060】

この本発明は、ここに説明され機能を実現化し、また添付されたフローチャートに示されているコンピュータ・プログラムを含む。しかしながら、本発明のコンピュータ・プログラムによる実現には多くの異なるやり方があり得ること、また本発明はどれか1組のコンピュータ・プログラム命令に限定された形で構成されるべきでないことは理解されるであろう。さらに、熟練したプログラマは、例えばこの出願書中のフローチャートや関連した説明に基づいて、開示された発明を実現するためのプログラムを困難なしに書くことができるであろう。したがって、ある特定の1組のプログラム・コード命令による開示は、本発明の実現の仕方および使い方の十分な理解のために必要であるとは考えられない。特許請求されているコンピュータ・プログラムの発明をなす機能は、プログラムのフローを示す残りの図と共に下記の説明において、より詳しく説明されるであろう。

【0061】

1つの例としての実施の形態では、フュージョン・エンジン22はオブジェクト指向のプログラムで実現される。したがって、図6に示されるソフトウェア・コンポーネントの幾つかは、それぞれのソフトウェア・オブジェクトに関連するデータおよびコードの両方を持ち得る。しかしながら、各ソフトウェア・オブジェクトの一般的な機能は、熟練したプログラマが、そのソフトウェア・オブジェクトの開示された機能を実現するためのコンピュータ・プログラムを書くことができるであろうように、一般的に説明されている。

【0062】

フュージョン・エンジン22は幾つかのソフトウェア・コンポーネントを含むことができる。図6に示されている説明のための実施の形態では、フュージョン・エンジン22はイベント収集部24からの生のコンピュータ・イベント情報を受け取るイベント読取り部600を含んでもよい。イベント読取り部600は、分類部615に動作によりリンクされる。分類部615はイベント読取り部600から受け取った生イベントの情報を整理する。換言すれば、分類部615は各生イベントに含まれるイベント・タイプ・プロパティに従って生イベント情報を仕分けすることにより生イベント情報を分類する。各生イベントのイベント・タイプ・プロパティは通常は侵入探知システム内のディテクタによって生成される。

【0063】

分類部615は、CoBRAプロセッサ625および1つまたはそれ以上の相関ルール620に対して生のイベント情報を送る責任を持つこともできる。1つまたはそれ以上の相関ルール620は、セキュリティに係わる出来事が発生しつつある可能性があるかどうかテストし判断するためのアルゴリズムを含むことができる。相関ルールは分類部から受け取った生のイベント情報を追跡し、その生のイベント情報を相関イベント高速メモリ665に記憶する。相関イベント高速メモリ665は、情報を記憶するためのランダム・アクセス・メモリ(RAM)から構成されてもよい。しかしながら、本発明はRAMタイプのメモリに限定されない。他の高速メモリ・デバイスも本発明の範囲を逸脱しない。分類部615は、メモリイベント分類データベース635に基づいて創設されることができる。分類部615は、フュージョン・エンジン22の初期化時にイベント分類データが生イベント分類データベース635から分類部615に読み込まれるときに生成されることができる。

【0064】

CoBRAプロセッサ625は、生のコンピュータ・イベントの状況に基づきリスク調整のためのアルゴリズムまたはソフトウェア・コンポーネントを含んでもよい。CoBRAプロセッサ625は、生のコンピュータ・イベントを、状況または知識ベースのデータベース630内に含まれているデータと比較することにより、生のコンピュータ・イベントの優先値を調整することができる。生イベントの優先順位は、通常は、その生イベント・データをフュージョン・エンジン22に転送する前に、侵入探知システムのディテクタによって設定される。生のコンピュータ・イベントを処理した後、フュージョン・エンジン

10

20

30

40

50

22は、セキュリティ・イベントが発生しつつあるかどうかを、イベント・コレクタ24に知らせることができる。フュージョン・エンジン22は、通常は、1つまたはそれ以上の関連イベントをフォーマットし、イベント・リポータ660を介してイベント・コレクタに送る。上で述べたように、関連イベントは、フュージョン・エンジン22によって決められて互いに関係づけられた1つまたはそれ以上のコンピュータ・イベントを含んでもよい。

【0065】

フュージョン・エンジン22はさらに、フュージョン・エンジン22のためのメモリ資源を浪費しないためのメモリ管理デバイスを含んでもよい。例えば、1つの例としての実施の形態では、フュージョン・エンジン22は、メモリ管理リスト640、生イベント追跡インデックス645および成熟イベント・リスト650を含んでもよい。メモリ管理リスト640は通常は生イベント追跡インデックス645にリンクされる。メモリ管理リスト640、生イベント追跡インデックス645および成熟イベント・リスト650に関する機能のさらに詳しいことは、図6に示されるソフトウェア・コンポーネントの簡潔な処理の説明において以下に説明されるであろう。

【0066】

図6のための説明のためのオブジェクト指向アーキテクチャ

1つの例としての実施の形態における、ソフトウェア・オブジェクトとして実現できる、フュージョン・エンジン22のソフトウェア・コンポーネントの1つは、イベント・リダー600である。イベント・リダー600は、イベント・コレクタ24またはイベント・ログ・ファイル610から生のコンピュータ・イベントを受け取ることができる。イベント・リダー600は、侵入探知システムからのコンピュータ・イベント・データを記憶するための、コンマで値(Comma Separated Value: CSV)を分けたフォーマットを持つファイルを含むことができる。イベント・リダー600は、通常は、生のコンピュータ・イベント、すなわち1つまたはそれ以上のコンピュータに対する攻撃である可能性があるものとして侵入探知システムによって追跡され得るかも知れないすべてのコンピュータ活動であり得る生のイベントを読み込む。このイベント・リダーは通常フュージョン・エンジン22上で他のソフトウェア・コンポーネントによって処理される生イベント・データ・オブジェクトを生成する。

【0067】

1つの例としての実施の形態では、1つまたはそれ以上のイベント・タイプ・オブジェクトを含んでもよいイベント・リダー600は分類部615にリンクされることができる。分類部615はイベント・リダー600によって生成された生イベントのオブジェクトを受け取る。分類部615は各生イベント・オブジェクトを、それに対応する、特定のイベント・タイプのパラメータ555に対して設定されているイベント・タイプのオブジェクトに関係づける。換言すれば、分類部は、生イベントのタイプに従って、生イベント・オブジェクトをイベント・タイプ・オブジェクトに指定する。イベント・リダー600によって受け取られる各生イベントは、その生イベントを生成した侵入探知システムに基づいてタイプまたはカテゴリに指定される。

【0068】

分類部615の1つの機能は、生イベントの各々を分類またはクラス分けし、そしてそれの生イベント・オブジェクトを、それらのタイプに基づいて特定の関連ルール620に送ることである。関連ルール620は、分類部615から生イベントを受け取るソフトウェア・オブジェクトの形をとることもできる。

【0069】

分類部615は、コンテキスト・ベースド・リスク・アジャストメント(CoBRA)プロセッサ625に生イベント・オブジェクトを送ることもできる。CoBRAプロセッサは、生イベント・オブジェクトの優先度パラメータを調整することができるリスク評価の仕組みである。CoBRAプロセッサ625は、受け取った生イベント・オブジェクトの各々に対して状況に基づくリスク調整を行うために、状況または知識ベースのデータベ

10

20

30

40

50

ス630にアクセスする。基本的には、C o B R A プロセッサは、攻撃の発源に加えて攻撃の宛先インターネット・プロトコル・アドレスのような環境因子との組合せでイベント・タイプ・パラメータ555を評価することにより、生コンピュータ・イベントのリスクを判断する。

【0070】

状況または知識ベースのデータベース630は、ネットワーク内のマシンに指定されている脆弱性、ヒストリカル・イベント頻度値およびネットワーク・ゾーン設定を含むことができる。脆弱性は、攻撃に対するネットワークまたは単一のコンピュータの強さすなわち抵抗力を判断するためにフュージョン・エンジン22の外部のデバイスにより行われる脆弱性スキャンの結果であってもよい。ヒストリカル・イベント頻度値は、署名すなわち非常に長い時間の間に発生したコンピュータ攻撃にデータを含むことができる。ネットワーク・ゾーンの定義は、ネットワークのある部分においてアクセスできるかも知れない情報の量と質に基づいてネットワークの各部分に付けられた値を含むことができる。例えば、以下に説明されるであろうように、内部、外部、および非武装化されたゾーンを区別することは役に立つ。

【0071】

フュージョン・エンジン22はさらに、分類部615を形成する異なるイベント・タイプのオブジェクトを生成する責任を持つことができる生イベント分類データベース635を含むことができる。生イベント分類データベース635は1つまたはそれ以上のセキュリティ情報の表を含むことができる。これらの表は、ディテクタによって付けられた、生イベントのタイプ・パラメータ555に関連する情報を含むことができる。生イベント分類データベース635は、生イベントを、標的にされたホストに対する衝撃（秘密保持性、完全性または利用可能性）、それらの範囲（ネットワーク、ホストまたはサービス）、およびそれらが用いる方法（裏口侵入、隠れ表、その他）に基づいて、生イベントを分類することができる。秘密保持性イベントは、攻撃者がホストからまたはホストについての情報を得ようと試みていることを示すイベントであり得る。完全性イベントは、攻撃者がホスト上のデータを変えようとしている、もしくはすると許可されないアクセスをしようとしていることを示すイベントであり得る。

【0072】

利用可能性イベントは、攻撃者が、例えばホストをクラッシュさせることによって、サービスの拒否を引き起こそうと試みていることを示すイベントであり得る。上記の一般的な基準に加えて、特定の相関イベントを見つけ出すのに役に立つ特別な基準が、イベントを分類するための基礎の役目を果たす。例えば、サービスの拒絶の試みの成功を確認するイベントは、成功したと信じられているサービス拒否攻撃を識別する相関ルール620によって使われるカテゴリにまとめられる。しかしながら、生イベント分類データベース635は、これらのカテゴリまたはパラメータに限定されない。さらに生イベントを定める他のカテゴリやパラメータも、本発明の範囲を逸脱しない。

【0073】

フュージョン・エンジン22はさらにメモリ管理リスト640、生イベント追跡インデックス645、および成熟イベント・リスト650を含むことができる。メモリ管理リスト640は、フュージョン・エンジン22が、メモリ資源が予め決められた閾値を越えたとき（すなわち、メモリ資源が少なくなったとき）最も古い生イベントを消去または削除することにより、そのメモリ資源を管理することを可能にする。メモリ管理リスト640は、メモリ資源が少なくなったとき最も古いと考えられる生イベントを削除するソフトウェア・オブジェクトとして実現することができる。メモリ管理リスト640に關係づけられているのは、同じくもう1つのソフトウェア・オブジェクトとして実現できる生イベント追跡インデックス645である。生イベント追跡インデックス645は、ある特定の生イベントプロジェクトを含むかも知れないソフトウェア・オブジェクトを特定することができる。すなわち、生イベント追跡インデックスは、今や古くなってフュージョン・エンジン22から削除されるべき生イベントを記憶しているかも知れないソフトウェア・オブジェ

10

20

30

40

50

クトを示す。

【0074】

メモリ管理リスト640と生イベント追跡インデックス645には、メモリ管理リスト640から除去されるべきでない活動パターンすなわち実際のコンピュータ脅威と認定された生イベントを追跡する成熟関連イベント・リスト650が関係づけられる。換言すれば、成熟関連イベント・リストは、成熟関連イベントすなわち実際のコンピュータ・セキュリティ脅威の一部であると見なされているのでフュージョン・エンジン22から削除されるべきでない生イベントを示している。

【0075】

フュージョン・エンジン22はさらに、各ソフトウェア・オブジェクトの間のデータ・フローの責任を持つコントローラ655を含んでもよい。換言すれば、コントローラ655は、それより下位のレベルのソフトウェア・オブジェクトの間のデータ・フローを制御するハイ・レベルのソフトウェア・オブジェクトとして実現できる。

【0076】

フュージョン・エンジン22はさらに、説明のためのそして好ましいオブジェクト指向プログラミング環境におけるソフトウェア・オブジェクトとして実現されることもできるイベント・リーダ600を含んでもよい。イベント・リーダ600は、イベント・コレクタ24に回送される成熟関連イベントを受け取る1つのソフトウェア・オブジェクトであってもよい。成熟関連イベントは、互いに関係づけられた1つまたはそれ以上の生イベントを含むことができる。なぜなら、その1つまたはそれ以上の生イベントが実際のコンピュータ・セキュリティ脅威である可能性があるからである。

【0077】

セキュリティ情報を管理するためのコンピュータにより実現されたプロセス

次に図7を参照する。この図は、1つまたはそれ以上のデータ・ソースから収集されたセキュリティ情報を管理するためのコンピュータにより実現されたプロセスの説明のための論理流れ図を示す。より具体的には、図7に示されている論理流れ図は、1つまたはそれ以上のコンソールに整理された情報の提示を行うために、複数のデータ・ソースから受け取ったセキュリティ情報を融合するすなわち組み合わせてそのセキュリティ情報を分析するためのコンピュータにより実現されたプロセスを示す。図7に説明される論理の流れは、フュージョン・エンジン22の最高レベルの処理ループの核心の論理であり、フュージョン・エンジン22が動作しているかぎりそれ自体繰り返し実行される。

【0078】

図7に示される論理流れ図は、図6に示されるソフトウェア・コンポーネントのうちの幾つかの初期化の後に発生するプロセスを示す。すなわち、本発明の説明のためのオブジェクト指向プログラミング・アーキテクチャにおいては、図7に示されるステップを実行するために必要とされるソフトウェア・コンポーネントすなわちソフトウェア・オブジェクトの幾つかが、図7によって説明されるプロセスの前に初期化されるかまたは生成される。したがって、この分野の通常の専門家は、図6に示されるソフトウェア・オブジェクトの初期化に幾つかのステップが示されていないことがわかるであろう。例えば、上に説明するように、分類部615を含むソフトウェア・コンポーネントすなわちソフトウェア・オブジェクトは、フュージョン・エンジン22の初期化の後に設立される。

【0079】

フュージョン・エンジン22の初期化中に、分類部615は、生イベント分類データベース635から情報を読み込むことにより構築される。分類部615はフュージョン・エンジン22によって処理されることができる生イベントのタイプに対応するイベント・タイプ・オブジェクトの包括的なリスト、および生イベント分類データベース635に定義されている各イベント・カテゴリのための別個のイベント・タイプ・オブジェクトのリストを含んでもよい。各別個のイベント・タイプ・リストは、生イベント分類データベース635によって、1つのカテゴリに属するものとして定義される1組の生イベントのタイプを構成する、包括的なイベント・タイプ・リストのサブセットを含むことができる。図6

に示されるいろいろなソフトウェア・コンポーネントの初期化は具体的に説明されていないが、熟練したプログラマは、この出願におけるソフトウェア・アーキテクチャの下記のフローチャートおよび関連する説明に基づいて、困難なく、開示された発明を実現するためにこのようなコンピュータ・プログラムを書くことができるであろう。

【0080】

以下に説明されるプロセス中のあるステップは、説明されるように機能するためには、本発明の他のものよりも本来的に先に実行されなければならない。しかしながら、本発明は、このような順番または順序が本発明の機能を変えないならば、説明されているステップの順番に制限されない。すなわち、いくつかのステップは、本発明の範囲と趣旨から逸脱することなく、他のステップの前または後に実施されてもよいことが、理解されるであろう。

【0081】

再び図7を参照する。この図は、コンピュータ・セキュリティ管理プロセスの最高レベルの処理ループの中核な論理の概観を与える。判断ステップを705において、フュージョン・エンジン22によって処理されるべき生イベントがあるかどうか判断される。上述したように、生イベントは、侵入探知システムのディテクトから報告されるコンピュータ・イベントを含んでもよい。侵入探知システムによって識別された生のコンピュータ・イベントは、いろいろなパラメータを含んでもよい。例えば、1つの例としての実施の形態では、各生イベントは、送信元のインターネット・プロトコル・アドレス、宛先のインターネット・プロトコル・アドレス、報告されているコンピュータ・イベントのタイプ、優先度、脆弱性およびタイムスタンプを含んでもよい。

【0082】

問い合わせで判断するステップ705が否定ならば、プロセスがステップ785に進む「ノー」枝が迫られる。問い合わせで判断するステップ705が肯定のときは、「イエス」枝が辿られてステップ710に進み、そこで生のコンピュータ・イベントまたはイベント情報がデータ・ソースから引き出される。データ・ソースは、図8に示されるように、イベント・データベース26、イベント・ログ・ファイル610、またはイベント・コレクタ24の少なくとも1つを含んでもよい。

【0083】

図8を一時的に参照する。この図は、図6に示されているいろいろなソフトウェア・コンポーネントの間の情報の交換を示すデータ・フロー図である。図8のこのデータ・フロー図は、図7に示されているステップと並行している。例えば、データ・ソースからイベント情報を引き出すためのステップ710が図8に示されているが、図8では説明のためのオブジェクト指向ソフトウェア・アーキテクチャ中のイベント・リーダー・オブジェクト600がイベント情報を読み込む。図8の参照は、図7の詳細な説明を全体を通してなされるであろう。

【0084】

再び図7を参照すると、ステップ710の後そしてステップ715において、イベント情報すなわち生イベントは、生イベントと呼ばれる所定のフォーマットに整理される。換言すれば、説明のためのオブジェクト指向プログラミング環境において、イベント・リーダー・オブジェクト600は、イベント・データベース26、イベント・コレクタ24、およびイベント・ログ・ファイル610のようなデータ・ソースの1つから生イベントを受け取ったとき、各生イベントに対してソフトウェア・オブジェクトを生成することができる。イベント・リーダー600は、コントローラ655から受け取ったコマンドに応じて生イベント・オブジェクトを生成する。換言すれば、コントローラ655が、イベント・リーダー600に、データ・ソースの各々から生イベントを引き出すように要求する。

【0085】

ステップ715の後、ルーチン720において、各生イベントからのイベント・タイプが確認され、各生イベントはイベント・タイプ・リストの中の対応するイベント・タイプ・オブジェクトに割り当てられる。換言すれば、説明のためのオブジェクト指向ソフトウェア

10

20

30

40

50

ア・アーキテクチャにおいて、イベント・リーダ600によって生成される各生イベントのオブジェクトは、クラシファイヤ615内に存する対応するイベント・タイプ・オブジェクトに送られる。ルーチン720のさらに詳しいことは、図9を参照して説明されるであろう。

【0086】

次に、判断ステップ725において、コンテキスト・ベース・リスク調整プロセス(CoBRA)625が起動されているかいないかを断定する。換言すれば、ユーザは、各生イベント内にある優先度情報のどれをも調整しないように決めることができる。上で説明したように、侵入探知システム内のディテクタによって生成された各生イベントは、そのイベントの優先度に設定されたパラメータを含む。すなわち、侵入探知システムのディテクタは、生イベントに付随しているかもしれない危険または起こり得る被害を評価するために、コンピュータ・イベントに相対値を付ける。例えば、ネットワークに対する分散された攻撃は、単一のマシンまたはコンピュータに対するコンピュータ攻撃に比べて、より高い優先順位を与えられるようにすることができる。

【0087】

問い合わせる判断するステップ725が否定のときは、「ノー」枝が辿られてルーチン740に進む。問い合わせる判断するステップ725が肯定のときは、「イエス」枝が辿られてステップ730に進み、そこで生イベントのパラメータが状況または知識ベースのデータベース630中の情報と比較される。またこのルーチンの中で、コンテキスト・データベース630中に在るコンテキスト情報に基づいて、各生イベントにコンテキスト・パラメータが付けられる。しばらく図8を参照すると、イベント・タイプ・オブジェクトを含むクラシファイヤ615は、CoBRA処理オブジェクトまたはCoBRAプロセッサ625に各生イベントを送送する。ルーチン730において、CoBRAプロセッサ625は、生イベントの環境または周辺条件に係るコンテキスト・パラメータを付けることができる。

【0088】

ルーチン730に続いて、ルーチン735の中で、各生イベントの優先順位は、CoBRAにより与えられたコンテキスト・パラメータまたはディテクタにより与えられたタイプ・パラメータまたはその両方に基づいて調整されるようにすることもできるし、元の順位がそのまま残されるようにすることもできる。基本的にはルーチン730および735は、CoBRAプロセッサ625の説明のためのアルゴリズムや方法を含むことができる。ルーチン730および735のさらに詳しいことは、図10、11および12に関して以下に説明されるであろう。

【0089】

次に、ステップ737において、CoBRAによって処理された生イベント、または処理されなかった生イベントは、イベント・コレクタ24のような、出力デバイスに送られることができる。イベント・コレクタ24は次に通常CoBRAにより処理された生イベントまたは処理されない生イベントを、イベント・データベース26に記憶し、そして次にそのイベントをコンソール30に送送する。以下の説明から明らかになるであろうように、コンソール30には、処理されていない生イベント、CoBRAにより処理された生イベント、および相関イベントが供給される。すべてのこのようなイベントはフュージョン・エンジン22によって処理されることができ、そしてユーザに対して表示されることができるようになる。イベント・コレクタ24に送られる。1つの生イベントがイベント・コレクタ24によってデータ・ソース28から受け取られたとき、イベント・コレクタはまずその生イベントをフュージョン・エンジン22に送ることが注意される。しかしながら、もし所定時間後にフュージョン・エンジン22が応答しない場合は、イベント・コレクタ24はそのイベントをイベント・データベース26に記憶し、それから処理されていない(フュージョン・エンジン22によって処理されていない)生イベントをコンソール30に送るであろう。

【0090】

10

20

30

40

50

ルーチン 740 において、生イベントは、ディテクタ 28 によって付けられたイベント・タイプに基づいて、相関ルール 620 と結合される。このルーチンでは、イベント・タイプ・オブジェクトを含むクラシファイヤ 615 は、どの相関ルール（1 つまたはそれ以上の）620 がそのイベント・タイプ・パラメータ 555 に基づいてその生イベントを処理すべきかを判断する。ルーチン 740 のさらに詳しいことは、図 13 に関して以下に説明されるであろう。

【0091】

判断ステップ 745 において、生イベントに合致するルールが存在する場合には、その相関ルールに関係づけられた相関イベントが存在するかどうか判断が下される。図 7 では単一のプロセス・フローとして表わされているけれども、ステップ 745 から 780 は、実際には、生イベントと関係づけられた各相関ルール 620 に対して独立に行われる。基本的には、判断ステップ 745 において、相関ルール・オブジェクトまたは相関ルール 620 が、処理されつつある現在の生イベントに対して相関イベント・オブジェクトが生成されたかどうか判断する。図 8 に示されるように、相関ルール・オブジェクトまたは相関ルール 620 は、相関ルール・オブジェクトまたは相関ルール 620 は、相関イベント・キャッシュまたは相関イベント高速メモリ 665 を調べて、処理されつつある現在の生イベントのための相関イベントが生成されたかどうか判断する。上で説明したように、相関イベント（またはオブジェクト指向ソフトウェア・アーキテクチャでは相関イベント・オブジェクト）は、単一のより高レベルのイベントを形成するために一緒にまとめられる複数の生イベントを含むことができる。

【0092】

ステップ 745 のために、各相関イベントは、相関イベント・キャッシュ 665 内の相関イベント・タイプの領域中にその相関イベントの索引を作るために使用される、アンカのインターネット・プロトコル（IP）アドレスを持つ。このアンカ IP アドレスは、その相関イベントのタイプに依って、その相関イベント内の 1 つまたはそれ以上の生イベントの送信元 IP アドレスまたは宛先 IP アドレスであろう。例えば、攻撃を受けたホストからの攻撃イベントの IP アドレスは、攻撃を受けたホストの IP アドレスである。これは外来する攻撃の宛先アドレスおよび出ていくアドレスの送信元アドレスである。攻撃を受けたホストからの攻撃イベントに対する相関ルールは、それに対する生イベントが外来攻撃であるであろう相関イベントを引き出すことを試みるときの相関イベント探索キーとして、外来する生イベントの宛先 IP アドレスを使う。AFH 相関ルールは、それに対する生イベントが出ていく攻撃であるであろう相関イベントを引き出すことを試みるときの相関イベント探索キーとして、生イベントの送信元 IP アドレスを使う。

【0093】

問い合わせて判断するステップ 745 が肯定のときは、ステップ 760 に向かって「イエス」枝が辿られる。問い合わせて判断するステップ 745 が否定のときは、ステップ 750 に向かって「ノー」枝が辿られ、そこで現在の相関ルールと結合された、予め決められたタイプの相関イベントが生成される。すなわち、この説明のためのオブジェクト指向ソフトウェア・アーキテクチャでは、生イベントの結合された相関ルール 620 の処理の中のこの点において、1 つまたはそれ以上の相関イベント・オブジェクトが生成され得る。

【0094】

次に、ステップ 755 において、この相関イベントは高速メモリ・デバイス 665 に記憶される。この説明のための実施の形態の高速メモリ・デバイスは、ランダム・アクセス・メモリ（RAM）を含むことができる。しかしながら、他の高速メモリ・デバイスも本発明の範囲を逸脱しない。現在のネットワークの処理速度と対応する情報量のために、生情報の迅速な処理を可能にするため RAM のような高速メモリ・デバイスを使用することが必要である。

【0095】

ステップ 760 において、生イベントは、生イベントのタイプに基づいて、対応する相関イベント（ステップ 750 において生成されたばかりのものか、またはステップ 745 に

10

20

30

40

50

において相関イベント・キャッシュ665から引き出されたものどちらでもよい)と結合される。換言すれば、説明のためのオブジェクト指向ソフトウェア・アーキテクチャ中のこのステップにおいて、各相関ルール・オブジェクトは、そのタイプに基づいて、生イベントを格納する。生イベントを相関イベントと結合することに加えて、生イベント追跡インデックス645が、その生イベントは相関イベントと結合されていることを示すために更新される。

【0096】

次に判断ステップ765において、処理されている現在の相関イベントがすでに成熟しているかどうか判断される。通常は、成熟しているためには、その特定のタイプの相関イベントに対して決められた成熟性の基準を満たす2つまたはそれ以上の生イベントを含むことができる。相関イベントの各タイプに対する成熟性の基準は、2つまたはそれ以上の生イベントの発生が、起こりそうなセキュリティ事件が起こりつつあることを示す条件を明らかにするために規定されている。ステップ765では、この相関イベントは、先行生イベントの処理の結果としてすでに成熟していると見なされているかどうか決めるために調べられている。

【0097】

問い合わせで判断するステップ765が肯定のときは、ステップ780に向かって「イエス」枝が辿られる。問い合わせで判断するステップ765が否定のときは、ルーチン770に向かって「ノー」枝が辿られる。ルーチン770において、処理されている生イベントが新しく結合された現在の相関イベントが、1つまたはそれ以上の相関ルール620として記述された成熟性の基準を満たすかまたはそれに合っているかどうか判断される。ルーチン770において、処理されている生イベントに対応するルーチンの各々が、現在の相関イベント中にリストされている現在の生イベントおよび他の生イベントがそのルールで記述されている成熟性の基準と一緒に満足するかどうか判断する。本発明は、与えられた生イベントのタイプ・パラメータに対応する任意の数のルールを含むことができる。

【0098】

1つの例としての実施の形態では、フュージョン・エンジン22は多数の相関ルール620を用いることができる。相関ルールは、悪意がある活動または悪意がない活動のどちらかを示すイベント・パターンを識別するための基礎として、生イベント分類データベース635に規定されているイベント・カテゴリを使うことができる。相関イベントおよび相関ルールの多くのものが、攻撃者の意図を明らかにすることができる。本発明によって検出される1組の相関イベントおよび対応する相関ルールは下記のものを含むことができる、ただしそれらに限定されない。

【0099】

1) 攻撃を受けたホストからの攻撃。このイベントは、あるホストに対する完全性攻撃に続いてそのホストから秘密性、完全性、または利用可能性攻撃が出されるのが見られたときに生成されることができる。

【0100】

2) 利用可能性攻撃スweep (マルチホストDOS攻撃)。このイベントは、同じ送信元IPアドレスから出た2つまたはそれ以上のタイプの利用可能性攻撃が複数の標的IPアドレスに対して見られたとき生成されることができる。

【0101】

3) 秘密性攻撃スweep (マルチホスト情報収集)。このイベントは、2つまたはそれ以上のタイプの秘密性攻撃が、単一の送信元IPアドレスから、複数の標的IPアドレスに対して、出ているのが見られたとき生成されることができる。

【0102】

4) DOSとそれに続く確認イベント。このイベントは、ある標的IPアドレスに対する利用可能性攻撃に続いてその標的がもはや正常に動作していないことを示すもう1つのイベントが見られたときに生成されることができる。確認イベントは、ホストが連絡不能であることを示す、ネットワーク上に設けられたセンサによって検出(例えば、他のホスト

10

20

30

40

50

からのその標的に対する A R P 要求の検出)されるイベント、およびホスト上に設けられたセンサによって標的とされたシステムそれ自体上で検出された、システムの資源(メモリのような)が枯渇したことを示すイベントを含む。

【0103】

5) 内部 I P アドレスを用いた外部送信元。このイベントは、外部ネットワークを監視するネットワーク上に設けられたセンサがそっくり同じ内部 I P アドレスを検出したときに生成されることができる。この状態の発生は、外部のホストが内部のホストの I P アドレスを使うこと、スプーフィング(だますこと)として知られているやり方、を試みていることを示す。

【0104】

6) 完全性攻撃とそれに続くリモート・ログイン。このイベントは、あるホストに対する完全性攻撃に続いて、そのホストからリモート・ログインが出されるのが見られたときに生成されることができる。

【0105】

7) 完全性攻撃とそれに続くサービスの開始。このイベントは、あるホストに対する完全性攻撃に続いて、ホスト上に設けられたセンサからのそのホスト上で新しいサービスが始まったという報告があったときに生成されることができる。

【0106】

8) インターネット・スキャン・スキャン。このイベントは、I S S インターネット・スキャンのスキャンが1つのホストから検出されたときに生成されることができる。スキャンの開始の検出に続くある期間の間は、その同じホストから発出する全ての他のイベントはインターネット・スキャン・スキャン・イベントに含められる。送信元 I P アドレスが承認されたスキャン・ソースとして構成されている場合には、そのイベントは悪意の無いイベントとみなされ得る。そうでない場合にはそのイベントは悪意のあるイベントとみなされ得る。

【0107】

9) 調査(プローブ)とそれに続く完全性攻撃。このイベントは、あるホストに対する調査イベントに続いて、そのホストに対する完全性攻撃が見られたときに生成されることができる。

【0108】

10) 完全性攻撃スイープ(犠牲者を捜し回ること)。このイベントは、2つまたはそれ以上のタイプの完全性攻撃が単一の送信元 I P アドレスから複数の標的 I P アドレスに対して発出するのが見られたときに生成されることができる。

【0109】

11) D o S 攻撃されたホストからのログイン。このイベントは、進行中の利用可能性攻撃の標的に現在になっている送信元 I P アドレスからのリモート・ログインが見られたときに生成されることができる。イベントのこの組合せは、攻撃者が、ネットワークの信頼関係を利用してネットワーク上の他のマシンにアクセスするためにあるホスト(利用可能性攻撃の標的)に変装していることを示している可能性がある。

【0110】

12) 複数のホスト上での1ユーザのログインの失敗。このイベントは、同一のユーザのログインの失敗が、ネットワーク上またはホスト上の複数のセンサによって報告されたときに生成されることができる。

【0111】

13) 疑わしい活動とそれに続く利用可能性攻撃。このイベントは、利用可能性攻撃が後に続く、クローキング法を含むイベントが報告されたときに生成されることができる。「クローキング」という語は、侵入探知システムから攻撃を隠そうとする全てのテクニックに適用される。

【0112】

14) 疑わしい活動とそれに続く完全性攻撃。このイベントは、完全性攻撃が後続した、

10

20

30

40

50

クロッキング法を含むイベントが報告されたときに生成されることができる。「クロッキング」という語は、侵入探知システムから攻撃を隠そうとする全てのテクニックに適用される。

【0113】

15) 疑わしい活動とそれに続く完全性攻撃。このイベントは、完全性攻撃が後続した、クロッキング法を含むイベントが報告されたときに生成されることができる。「クロッキング」という語は、侵入探知システムから攻撃を隠そうとする全てのテクニックに適用される。

【0114】

16) 継続する利用可能性攻撃(集中DOS攻撃)。このイベントは、2つまたはそれ以上のタイプの利用可能性攻撃が単一の送信元IPアドレスから単一の宛先IPアドレスに向けて送出されるのが見られたときに生成されることができる。

【0115】

17) 継続する秘密性攻撃(集中情報収集攻撃)。このイベントは、2つまたはそれ以上のタイプの秘密性攻撃が単一の送信元IPアドレスから単一の宛先IPアドレスに向けて送出されるのが見られたときに生成されることができる。

【0116】

18) 継続する完全性攻撃(集中押入り試み)。このイベントは、2つまたはそれ以上のタイプの完全性攻撃が単一の送信元IPアドレスから単一の宛先IPアドレスに向けて送出されるのが見られたときに生成されることができる。

【0117】

19) ウェブ・スキャン。このイベントは、1つのウェブ・サーバに向けられた多数のウェブに関連した攻撃がある期間内に検出されたときに生成されることができる。一連のURLが調べられるようなウェブに関係する攻撃の特徴を調べることによって、ウィスカ(Whisker)のような特定のウェブ・スキャン・ツールの使用を特定することが可能であろう。

【0118】

本発明の範囲と趣旨から逸脱することなく、追加のルールを用いることもできる。ルーチン770のさらに詳しいことは、図14に関して以下にさらに詳しく説明されるであろう。しかしながら、図14に示されているルーチン770は1つのルールの適用を説明しているにすぎないことは特に言及される。図14示されている説明のためのルールは、上のリスト中の「攻撃を受けたホストからの攻撃」(AFH) 相関イベントに対応するルールである。攻撃を受けたホストからの攻撃の筋骨きも、図5Dから5Fに関して以下にさらに詳しく説明されるであろう。

【0119】

問い合わせで判断するルーチン770が否定ならば、「ノー」枝が辿られて判断ルーチン785に進む。問い合わせで判断するルーチン770が肯定のときは、「イエス」枝が辿られてステップ710に進み、そこで成熟イベント・メッセージが生成されて、イベント・コレクタ24のような出力デバイスに送られる。ステップ775において、イベント・レポータ660はその相関イベントが成熟であるという標識を受け取り、それからイベント・レポータ660はこのメッセージをイベント・コレクタ24に送る。

【0120】

ステップ780において、1つの生イベントが既に成熟している相関イベントに追加されるとき、相関イベント更新通知が出力デバイスに送られる。このステップで、イベント・レポータはその相関イベント更新通知をイベント・コレクタ24に送る。イベント・コレクタ24はイベント・データベース26の中のその相関イベントの表現を更新し、この情報をそこでユーザが視ることができるコンソール30に送る。このことは、ユーザが、進行中のセキュリティ事件(すなわち、1つの成熟相関イベント)の一部である追加の生イベントが発生したときに通知されることを可能にする。

【0121】

10

20

30

40

50

次に、判断ルーチン785において、成熟関連イベントのどれかが発生を止めたか否かを判断する。判断ルーチン785のさらに詳しいことは、図15に関して以下に説明される。

【0122】

問い合わせで判断するステップ785が否定ならば、「ノー」枝が辿られてステップ795に進む。問い合わせで判断するステップ785が肯定のときは、「イエス」枝が辿られてステップ790に進み、そこで1つの関連イベントが発生を停止したことを示すメッセージが送られる。このメッセージは、イベント・レポータ660からイベント・コレクタ24に送られることができる。次に、イベント・コレクタ24はイベント・データベース26内のすでに結論が出された関連イベントの表現を更新し、そしてこのメッセージをコンソール30に送る。

【0123】

ステップ795において、メモリ管理リスト中の最も古い生イベントおよび成熟関連イベントが消去されてもよい。フュージョン・エンジン22のメモリの量は限られているので、成熟する可能性が最も高い生イベントでメモリを満たしておく必要がある。フュージョン・エンジンは、メモリ管理リスト640、生イベント追跡インデックス645、および成熟イベント・リスト650のような幾つかのメモリ使用状況監視デバイスを持っている。1つの例としての実施の形態では、フュージョン・エンジンのメモリ使用状況監視デバイスはどれだけの量のメモリが利用可能であるか判断し、そしてメモリが容量まで満たされそうになると、メモリ使用状況監視デバイスは、使用可能メモリを増やすために、最も古い既存の記憶されている生イベントおよび成熟関連イベントを消去するであろう。成熟関連イベント内に含まれる生イベントは、メモリ管理リスト640から除去されるが消去はされない。生イベントが消去されるときはいつでも、その生イベントを含有するすべての未成熟関連イベントを見つけ出すために、生イベント追跡インデックス645が使われ、そしてその生イベントがそれらの未成熟関連イベントから取り除かれる。1つの未成熟関連イベントから生イベントが取り除かれてその未成熟関連イベントが他の生イベントを含まないときは、その未成熟関連イベントも消去される。

【0124】

次に図9を参照する。この図は、生イベントのタイプを見分けて各生イベントをクラスファイア615の対応するイベント・タイプ・オブジェクトにする図27のルーチン720のための、コンピュータで実現されたプロセスを示す。ルール720は、各生イベントがクラスファイア615中の対応するイベント・タイプと合致させられるステップ910で始まる。次に、ステップ915において、各生イベントのタイムスタンプが見つけられる。ステップ920において、各イベントが、ステップ915で見つけられたタイムスタンプに基づいて、メモリ管理リスト640に追加される。このリスト中の項目は、通常、上述したメモリ掃却処理中に最も古いイベントを見つけるのを容易にするために、タイムスタンプに従って順番に保たれる。ステップ925において、各生イベントは、クラスファイア615に含まれているようなイベント・タイプ・オブジェクトと結合された状態で、高速メモリに記憶される。次に、ステップ930において、生イベントを受け入れた各イベント・タイプ・オブジェクトが、生イベント追跡インデックス645に追加される。すなわち、通常、フュージョン・エンジンの各ソフトウェア・コンポーネントは、生イベントを受け取る、それ自身を生イベント追跡インデックス645に登録する。このように、ある生イベントがシステムから削除されることが決められたとき、生イベント追跡リスト645が、削除される必要のあるその生イベントの参照の場所を見つけるために使用できる。ステップ930の後に、このプロセスは図7の判断ステップ725に戻る。

【0125】

図10は、各生イベントのパラメータが状況または知識ベースのデータベース630と比較される図7のルーチン730のためのコンピュータにより実現されたプロセスを示す。このルーチンにおいても、コンテキスト・データベース630とこの比較に基づいて、各生イベントに対して追加のパラメータが付けられる。上述したように、コンテキスト・

10

20

30

40

50

データベース 630 は、生イベントの重要性を評価するのに役に立つかも知れない環境情報を含むことができる。例えば、コンテキスト・データベース 630 は、ネットワーク内のマシンすなわちコンピュータについての脆弱性情報、予め決められたゾーンに基づくコンピュータまたはディテクタの相対位置、およびヒストリカル・イベント頻度に関する情報を含むことができる。

【0126】

コンテキスト・データベース 630 の脆弱性情報は、通常、ネットワークを構成する 1 つまたはそれ以上のマシンに存在するかもしれない相対的なセキュリティに対する危険性を判断するためにネットワーク全体に対して行われるスキャンから導き出される。フュージョン・エンジン 22 によって監視されているネットワークのための、ヒストリカル生イベント・ログを解析するツールは、通常、コンテキスト・データベース 630 のヒストリカル・イベント頻度情報を導き出す。このツールは、通常、同じ生イベントのタイプ、送信元インターネット・プロトコル・アドレス、および宛先インターネット・プロトコル・アドレスを共有するイベント群に対してそれらの平均のイベント頻度を計算する、ただし平均イベント頻度を計算する目的のために生イベントをグループ分けするための他のやり方も本発明の範囲内にある。コンテキスト・データベース 630 のゾーンの定義は、通常、ネットワークの各部をそれらのネットワーク全体との関係にしたがって分類することにより導き出される。例えば、内部ゾーンおよび非武装化ゾーン (DBZ) は、内部ゾーンはインターネットからアクセス可能であってはいけいないネットワークのインターネット・プロトコル・ネットワーク・アドレスを含み、また DMZ ゾーンはインターネットからアクセス可能なネットワークのインターネット・プロトコル・ネットワーク・アドレスを含むようなものとして定義されることができる。これらのゾーンは、フュージョン・エンジン 22 によって監視されているネットワークのために適当に決められるであろう。

【0127】

ルーチン 730 は通常は CoBRA プロセッサ 625 によって行われる。CoBRA プロセッサ 625 は通常各生イベントを調べ、かつそれをコンテキスト・データベース 630 と比較する。より具体的に言えば、ステップ 1010 (ルーチン 730 の最初のステップ) において、宛先インターネット・プロトコル・アドレス情報およびコンテキスト・データベース 630 との比較に基づき、各生イベントに対して、CoBRA 脆弱度 504 が付けられる。1 つの例としての実施の形態では、付けられる脆弱性値は、脆弱であると思われる、脆弱でないと思われる、および不明、のどれか 1 つであることができる。

【0128】

次に、ステップ 1015 において、コンテキスト・データベース 630 とのもう 1 つの比較に基づき、各生イベントに対して、ヒストリカル頻度値 506 が付けられる。この値は、1 日あたりのイベントのような、単位時間当たりのイベントの数でもよいし、イベント間の平均時間のような、数学的に関係づけられる値でもよい。ヒストリカル・イベント頻度値は、通常、ある特定の送信元マシンからある特定の宛先マシンへのある特定のタイプの生イベントが、そのフュージョン・エンジン 22 によって監視されているネットワーク上で、どれくらいの頻度で見られるかを示す。ヒストリカル頻度データは、フュージョン・エンジンによって、正常な悪意の無いネットワーク活動によって引き起こされるイベントと異常な悪意のあるネットワーク活動によって引き起こされるイベントを識別するのを助けるためにフュージョン・エンジンによって使われる。

【0129】

ステップ 1020 において、各生イベントに対して、その生イベントの送信元インターネット・プロトコル・アドレスおよびコンテキスト・データベース 630 との比較に基づいて、送信元ゾーン 508 の値が付けられる。ステップ 1025 において、各生イベントに対して、各生イベントの宛先インターネット・プロトコル・アドレスおよびコンテキスト・データベース 630 との比較に基づいて、宛先ゾーン 510 の値が付けられる。

【0130】

ステップ 1030 において、各生イベントに対して、センサ・インターネット・プロトコ

10

20

30

40

50

ル・アドレスおよびコンテキスト・データベース 630 との比較に基づいて、センサ・ゾーン 512 の値が付けられる。センサ・ゾーン値は、疑わしいコンピュータの活動を検出してその生イベントを生成した侵入探知システムのセンサまたはディテクタのインターネット・プロトコル・アドレスを含むことができる。ステップ 1030 の後、プロセスは図 7 のルーチン 735 に戻る。

【0131】

次に図 11 を参照して、この図は、C O B R A により付けられたコンテキスト・パラメータ、またはディテクタにより付けられたタイプ・パラメータ、またはその両方に基づいて、生イベントの優先順位を調整するかまたは元の優先順位をそのまま残すことができる図 7 のルーチン 735 のためのコンピュータにより実現されたプロセスを示す。ルーチン 735 は C O B R A プロセッサ 625 のもう 1 つの中核機能である。このルーチンは、フュージョン・エンジンが、保護されているネットワークまたはコンピュータにとってのそれらの重要性に基づいて生イベントを順位づけることを可能にする。このようにして、セキュリティの管理者は、より効率的にそして効果的に、コンピュータ・セキュリティ・イベントを監視することができる。なぜなら重要なコンピュータ・セキュリティ・イベントは、他のより低順位のセキュリティ・イベントに対して相対的に高い順位および優先度を持つ。

【0132】

本発明は、それらのイベントおよびユーザにとって最も重要なネットワークの部分のために、ユーザにより定められた特性を使うことができる。例えば、コンテキスト・データベース 630 の一部を構成するゾーンの定義はユーザによって与えられてもよい。1 つの例としての実施の形態では、監視されているネットワークの内部ゾーンおよび所謂非武装化ゾーン (DMZ) はユーザによって設定されることができる。またユーザによって明示的に決められるよりもむしろ、外部ゾーンは、ユーザによって指定された 1 つの明示的に決められたゾーンまたは複数のゾーン内に入らないすべての I P アドレスであってもよい。本発明はこれらのタイプのゾーンに限定されない、そして例えばビジネス・パートナー・ゾーンのような他のタイプのゾーンを含むことができる。この分野の専門家は、本発明が、ユーザによって決められた任意の数のゾーンにインターネット・プロトコル・アドレスを割り当てるように設計され得ることを理解するであろう。

【0133】

上述したように、各生イベントは、侵入探知システム内のディテクタによってそれに付けられた優先順位パラメータ 535 を含む。1 つの例としての実施の形態では、この優先順位パラメータは次の 3 つの値、すなわち、1、2 または 3 のどれでも含むことができる。この説明のための実施の形態における最高の優先順位は、通常は数値 1 である。他方、最低の優先順位は、通常は値 3 である。中間の優先値は通常は数値 2 である。ディテクタによって付けられる優先順位値はその本質上非常に控えめなので、各生イベントのための優先順位値の調整が必要である。すなわち、生イベントは、通常、高いネットワーク・トラフィック・スピードを維持するためにディテクタ・レベルになければならない簡単な処理テクニックの結果である。

【0134】

従って、従来の侵入探知システムのディテクタ・レベルから来る優先順位値は、各イベント・タイプに対して適用できるであろう最悪の場合のシナリオに適切であるように決められる。例えば、この説明のための実施の形態では、あるタイプの生イベントがあるネットワーク上で当てはまるコンテキストに依って 1、2 または 3 の実際の優先度を持ち得るならば、ディテクタは通常このタイプのすべてのイベントに最悪の場合の優先度 (この説明のための実施の形態では (1) として表されている) を与えるであろう。C O B R A プロセッサ 625 がこの優先順位 535 の値を変更するときはいつでも、元のディテクタによって付けられた優先順位と優先順位 535 において更新されたすなわち C O B R A によって調整された優先パラメータの両方を記憶した後にだけそれを行う。すなわち、この説明のための実施の形態では、C O B R A 処理の後でかつ優先度が調整されたときに、調整

10

20

30

40

50

された優先順位 5 3 5 は、元のディテクトによって付けられ優先順位と C o B R A によって調整された優先順位の 2 つの値を含む。

【0135】

フュージョン・エンジン 2 2 は、環境条件すなわちその中で生イベントが発生される周囲状況に基づいて生イベントを順位づけることを可能にする。このようにして、セキュリティ担当のネットワーク管理者は、監視されているネットワークまたはコンピュータにとって最も重要なコンピュータ・セキュリティ・イベントだけを提示されるであろう。本発明は説明された優先順位階級に限定されない。すなわち、本発明は、1 が最高の優先度を割り当てられた 1 から 3 の範囲の優先階級に限定されない。他の範囲または値および値の間の増分も、本発明の範囲を逸脱しない。この分野の専門家は、重要でない生イベントが重要な生イベントよりも上位に順位づけられる可能性をさらに少なくするために、もっと複雑な階級を生成することができることがわかるであろう。

【0136】

ルーチン 7 3 5 はステップ 1 1 0 で始まり、このステップでは、生イベントの標的がそのコンピュータ攻撃に抵抗性があるかどうか判断される。この判断は、図 1 0 に説明されている手順 7 3 0 のステップ 1 0 1 0 によって先に設定された、その生イベントの C o B R A 脆弱状態 5 0 4 の値に基づいて行われる。問い合わせて判断するステップ 1 1 1 0 が否定ならば、「ノー」枝が辿られて、プロセスは図 1 2 のステップ 1 2 1 0 に進む。問い合わせて判断するステップ 1 1 1 0 が肯定のときは、「イエス」枝が辿られてステップ 1 1 1 5 に進む。ステップ 1 1 1 5 において、生イベントは、コンテキスト・データベース 6 3 0 内の 1 つのリストに記憶されている脆弱性が調整可能なイベント・タイプと比較される。コンテキスト・データベース 6 3 0 に記憶されているこれらの脆弱性が調整可能なイベント・タイプは、それらに対してはマシンの脆弱性順位の評価が信頼できると信じられており、したがって脆弱性順位情報に基づいて優先度を調整することが許されると、ユーザまたはシステムによって認定されたイベントである。

【0137】

そうする代わりに、もう 1 つの実施の形態（図示されていない）では、コンテキスト・データベース 6 3 0 が、その脆弱性順位の評価が信頼できるとユーザまたはシステムが信じる生イベント・タイプを識別することができ、そしてすべての他のイベント・タイプに対しては、脆弱性順位の評価は信じられると見なされ得る。このようにして、優先順位に関して調整されることが望まれない生イベントが識別されて、C o B R A プロセス 6 2 5 がこのような生イベントの優先度を引き下げないようにされる。もう 1 つの別の説明のための実施の形態（示されていない）では、コンテキスト・データベース 6 3 0 は両方のタイプのリストを含むことができる。すなわち、コンテキスト・データベース 6 3 0 は、調整されるべき優先順位を持つことが許される生イベント・タイプのリストと、調整される優先順位を持つことが許されない生イベント・タイプを含むリストを含むことができる。この場合には、あるイベント・タイプが両方のリストに現れたときにどちらの項が優先するか明確に定められるように、衝突回避ルールも設定されなければならない。この分野の専門家は、リストの他の構成も本発明の範囲を逸脱しないことを理解するであろう。

【0138】

次に、ステップ 1 1 2 0 において、コンテキスト・データベース 6 3 0 の記憶された脆弱性調整可能なイベントとの一致があるかどうか判断される。問い合わせて判断するステップ 1 1 2 0 が否定ならば、ステップ 1 1 3 5 に進む「ノー」枝が辿られる。問い合わせて判断するステップ 1 1 2 0 が肯定のときは、「イエス」枝が辿られてステップ 1 1 2 5 に進む。

【0139】

判断ステップ 1 1 2 5 において、処理されている現在の生イベントが最低の優先順位であるかどうか判断される。換言すれば、処理されている現在の生イベントが説明のための優先順位値 3 を持つならば、その優先度はそれ以上調整できないということがわかる。したがって、問い合わせて判断するステップ 1 1 2 5 が肯定のときは、「イエス」枝が辿られ

10

20

30

40

50

てステップ1135に進む。問い合わせで判断するステップ1125が否定のときは、「ノー」枝が辿られてステップ1130に進み、そこで現在の生イベントの優先順位535が引き下げられて、優先順位535の変更の理由がその生イベントに記録される。例えば、説明のための実施の形態では、生イベントが元の優先順位1を持ち、かつC O B R A プロセッサ625がそのイベントが脆弱であるとは思われないと判断したら、C O B R A プロセッサは、その元の優先順位値1を、値2（中間優先順位値）のようなより低い値に調整するであろう。優先順位値を変更する理由は、なぜある特定の生イベントが引き下げられた優先度を付けられたかコンソール30において分かるように、その生イベントの優先度変更理由516パラメータに記録される。1つの例としての実施の形態では、この生イベントの優先順位値を変更する理由は、文字列を含むことができる。

10

【0140】

ステップ1135において、各生イベントは、コンテキスト・データベース630内の1つのリストに記憶されている頻度調整可能なイベント・タイプと比較される。上で説明された脆弱度調整可能なイベント・タイプと同様に、頻度調整可能なイベント・タイプは、ある1対のマシンの間の高いヒストリカル・イベント頻度がフュージョン・エンジン22によって監視されているネットワークまたはコンピュータに対する悪意の無さの信頼できる指標として見られる生イベント・タイプを含むことができる。別の構成として、同じく上で説明された脆弱度調整可能なイベント・タイプと同様に、もう1つの例としての実施の形態（図示されていない）では、コンテキスト・データベース630は、その代わりに、フュージョン・エンジン22によって監視されているネットワークまたはコンピュータに対して、ある1対のマシンの間の高いヒストリカル・イベント頻度が悪意の無さの信頼できる指標として見られる生イベント・タイプを示すリストを含みことができ、かつヒストリカル・イベント頻度がすべての他のイベント・タイプに対しては悪意の無さの信頼できる指標と考えることができる。このようなシナリオにおいては、そのリストは、ヒストリカル・イベント頻度に基づいてその優先順位を調整することが望ましくない生イベントを示すであろう。代わりの構成として、さらにもう1つの例としての実施の形態（図示されていない）では、コンテキスト・データベース630は、両方のタイプのリスト、1つのリストは頻度に基づく優先度の調整が許される生イベント・タイプを示し、他方のリストは頻度に基づく優先度の調整が許されない生イベント・タイプを示す、を含むことができる。この場合には、あるイベント・タイプが両方のリストに現れたときどちらが優先されるか明確であるように、衝突解決ルールも決められなければならない。この分野の専門家は、リストの他の構成も本発明の範囲を逸脱しないことを理解するであろう。

20

30

【0141】

ステップ1135に続いて、ステップ1145において、評価されている現在の生イベントに対する一致が存在するか否か判断される。問い合わせで判断するステップ1145が否定のときは、図12のステップ1210に向かって「ノー」枝が辿られる。問い合わせで判断するステップ1145が肯定のときは、判断ステップ1150に向かって「イエス」枝が辿られる。

【0142】

判断ステップ1150において、評価されている現在の生イベントに対してヒストリカル頻度情報が存在するか否か判断される。この判断は、図10に説明されている手順730のステップ1015によって先に設定されたその生イベントのヒストリカル頻度値506に基づいて行われる。換言すれば、生イベントの中には、ヒストリカル頻度情報を得るために解析されるヒストリカル・データ中に見られないタイプ、送信元および宛先のものがある可能性がある。問い合わせで判断するステップ1150が否定のときは、図12のステップ1210に向かって「ノー」枝が辿られる。問い合わせで判断するステップ1150が肯定のときは、判断ステップ1150に向かって「イエス」枝が辿られる。

40

【0143】

判断ステップ1150において、評価されている現在の生イベントに対してヒストリカル頻度情報が存在するか否か判断される。この判断は、図10に説明されている手順730

50

のステップ1015によって先に設定されたその生イベントのヒストリカル頻度値506に基づいて行われる。換言すれば、生イベントの中には、ヒストリカル頻度情報を得るために解析されるヒストリカル・データ中に見られないタイプ、送信元および宛先のものがある可能性がある。問い合わせて判断するステップ1150が否定のときは、図12のステップ1210に向かって「ノー」枝が辿られる。問い合わせて判断するステップ1150が肯定のときは、判断ステップ1155に向かって「イエス」枝が辿られる。

【0144】

判断ステップ1155において、評価されている現在の生イベントに対するヒストリカル頻度が頻発イベント閾値よりも大きいかが判断される。換言すれば、この判断ステップでは、あるイベントが、特定の送信元と宛先の間に、悪意の無いイベントでありそうだと見なされ得るのに十分な頻度で発生するタイプであるかどうか判断される。頻発イベント閾値は、1日当たりのような、単位時間当たりの平均のイベント数に対応する値でもよい。しかしながら、他のこの値に数学的に関係づけられた形、例えばイベント間の平均時間、も使用することができ、本発明の範囲を逸脱しない。処理されている現在の生イベントが閾値よりも大きいヒストリカル・イベント頻度を持つときは、それは頻発イベントであり悪意の無いものであろうと見なされる。

【0145】

それが頻発・生イベントであると判断されたときは、その優先順位が引き下げられる。しかしながら、処理されている現在の生イベントがそのネットワーク上でより低頻度で見られなかった場合は、それは頻発・生イベントとはみなされず、そしてヒストリカル・イベント頻度に基づくその優先順位の調整は望ましくないとみなされる。したがって、問い合わせて判断するステップ1155が否定（処理されている現在の生イベントのようなイベントが、フュージョン・エンジン22によって監視されているネットワーク上で頻発には見られなかったことを意味する）のときは、図12のステップ1210に向かって「ノー」枝が辿られる。問い合わせて判断するステップ1155が肯定（処理されている現在の生イベントのようなイベントが、フュージョン・エンジン22によって監視されているネットワーク上で頻発に見られたことを意味する）のときは、判断ステップ1160に向かって「イエス」枝が辿られる。

【0146】

判断ステップ1160において、評価されている現在の生イベントが最低の優先順位であるかが判断される。この問い合わせて判断するステップ1160が肯定のときは、図12のステップ1210に向かって「イエス」枝が辿られる。この問い合わせて判断するステップ1160が否定のときは、ステップ1165に向かって「イエス」枝が辿られ、現在の生イベントの優先度が引き下げられ、現在の生イベントの優先順位を変更する理由が記録される。その理由は、通常は、評価されている生イベントが高頻度で発生するというように記録される。

【0147】

プロセスは次に図12に続く。図12は図7のルーチン735のためのコンピュータで実現されたプロセスの第2の部分を示し、それにおいて、CoBRAプロセッサ625が、その生イベントのCoBRAプロセッサ625によって付けられたコンテキスト・パラメータまたはディテクタによって付けられたタイプ・パラメータに基づいて、優先順位を調整するか、または元の優先順位をそのまま残す。

【0148】

ステップ1210において、生イベントは、コンテキスト・データベース630の1つのリストに記憶されているゾーンを調整可能なイベント・タイプと比較される。上で説明された脆弱性調整可能なイベント・タイプおよび頻度調整可能なイベント・タイプと同じように、ゾーンを変更可能なイベント・タイプは、ネットワークのセキュリティの管理者によって決められることができ、またそれらが内部で発生（すなわち、その生イベントの送信元インターネット・プロトコル・アドレスおよび宛先インターネット・プロトコル・アドレスの両方が、コンテキスト・データベース630において内部ゾーンに属すると定め

10

20

30

40

50

られているネットワーク上に在る)したとき、フュージョン・エンジン22によって監視されているネットワークまたはコンピュータに対して危険性が低いと見なされる生イベントのタイプである。しかしながら、別の実施の形態では(図示されていない)、コンテキスト・データベース630は、代わりに、送信元および宛先が位置するゾーンだけに基づいては、フュージョン・エンジン22によって監視されているネットワークまたはコンピュータに対して危険性が低いと見なされることができない生イベントのタイプを示すリストを含んでもよい。

【0149】

このような実施の形態においては、リストに記載されているもの以外のイベント・タイプは、それらが内部的に発生したとき、監視されているコンピュータまたはネットワークに対してリスクが低いと見なされる。さらに別の実施の形態(図示されていない)においては、コンテキスト・データベース630は、両方のタイプのリスト、すなわち、送信元および宛先のゾーンだけに基づいては危険性が低いと見なされることができず、その優先順位が調整されるべきではない生イベント・タイプを示す第1のリストと、内部で見られたときは危険が低いと見なされかつそれらのイベントがより低い優先順位を持つように優先順位が調整されるべきである生イベントの第2のリスト、を含んでもよい。この場合には、あるイベント・タイプが両方のリストに現れたときにどちらの項が優先するか明確に定められるように、衝突回避ルールも設定されなければならない。

【0150】

判断ステップ1215において、コンテキスト・データベース630内に記憶されているゾーンを調整可能なイベント・タイプとの一致が存在するか否か判断される。問い合わせで判断するステップ1215が否定のときは、図7のルーチン740に向かって「ノー」枝が後戻り方向に辿られる。問い合わせで判断するステップ1215が肯定のときは、判断ステップ1220に向かって「イエス」枝が辿られる。

【0151】

判断ステップ1220において、処理されている現在の生イベントの送信元ゾーンおよび宛先ゾーンが、両方とも、フュージョン・エンジン22によって監視されているネットワークまたはコンピュータに関して内部であるかどうか判断する。この判断は、それぞれ、図10に示されているルーチン730のステップ1020およびステップ1025によって与えられた生イベントの送信元ゾーンパラメータ508および宛先ゾーンパラメータ510の値を調べることによりなされる。多くのイベント・タイプに対しては、内部として分類された生イベントは、フュージョン・エンジン22によって監視されているネットワークまたはコンピュータに対して外部であるかもしれないイベントに比べて、監視されているコンピュータのネットワークに対する脅威の程度が低い。

【0152】

したがって、内部イベントに対しては、このような生イベントの優先順位を引き下げるものが望ましいかもしれない。反対に、送信元または宛先のどちらかのインターネット・プロトコル・アドレスがDMZゾーン内にあるかまたはどの定められたゾーン内にもないかのどちらかである(そして従って外部と見なされる)生イベントにたいしては、侵入探知システム中のディテクタによって与えられた生イベントの優先順位を維持することが望ましいかもしれない。問い合わせで判断するステップ1220が否定のときは、図7のルーチン740に向かって「ノー」枝が後戻り方向へ辿られる。問い合わせで判断するステップ1220が肯定のときは、判断ステップ1225に向かって「イエス」枝が辿られる。

【0153】

判断ステップ1225において、現在の生イベントがその最低の優先順位であるかどうか判断される。問い合わせで判断するステップ1225が肯定のときは、図7のルーチン740に向かって「イエス」枝が後戻り方向に辿られる。問い合わせで判断するステップ1225が否定のときは、ステップ1230に向かって「ノー」枝が辿られ、そこで現在の生イベントの優先順位が引き下げられ、そしてその生イベントの優先順位の変更の理由が記録される。通常、ステップ1230における理由は、現在の生イベントの優先順位が引き

10

20

30

40

50

下げられたのはそれが内部攻撃を構成するからであることを示すであろう。プロセスは次に図7のルーチン740に戻る。

【0154】

本発明はまた優先順位値の引き下げ方について限定されない。換言すれば、本発明は、引き下げられた優先度または引き上げられた優先度を表わすめの階級を含むこともできる。この分野の専門家は、どんな数の危険度調整スキームでも使用でき、本発明の範囲と趣旨から逸脱しないことを理解するであろう。

【0155】

次に図13を参照する。この図は、生イベントがイベント・タイプ・パラメータ555に基づいて予め決められた相関ルールと結合される図7のルーチン740のコンピュータにより実現されたプロセスを示す。このルーチンでは、クラシファイヤ615が、各々の与えられた生イベントを処理すべき1つまたはそれ以上の相関ルール620を見つけ出してよい。ステップ1310はルーチン740の最初のステップであり、このステップにおいて生イベントを含む全てのリストがCORA処理による変更を反映するために更新される。換言すれば、CORAプロセッサ625によって調整された生イベントを含む、説明のためのオブジェクト指向アーキテクチャ中の全てのオブジェクトが、優先順位すべての変更を反映させるために更新される。

【0156】

次に、ステップ1315において、生イベントが、その生イベントのタイプ・パラメータ555に適用される相関ルールに送られる。より具体的に言えば、ステップ1315において、各相関ルール620の定義は、それにとって関心のある生イベントのカテゴリの1つのリストを含む。各生イベント・カテゴリに含まれる生イベントのタイプは、生イベント分類データベース635中で定められている。したがって、ある相関ルール620に関心のある生イベント・タイプのリストは、そのルールにとって関心のあるカテゴリのための、カテゴリに特有の生イベント・タイプのリストの集まりであり、それにおいては各カテゴリに特有の生イベント・タイプのリストが生イベント分類データベース635によって定められている。カテゴリに特有の、生イベント・タイプのリストはクラシファイヤ615に記憶されており、生イベント分類データベース635の内容に基づいて初期化される。

【0157】

システムの初期化中にコントローラ655が相関ルール620をロードするとき、コントローラ655は、そのルールをそのルールにとって関心のあるイベント・タイプと結合させる。これはそのルールを各々のそのようなイベント・タイプ内に維持されている関係のあるルールのリストに追加することにより行う。したがって、初期化の後、各イベント・タイプは、そのタイプのイベントに関心を持つ相関ルール620の全てを列挙した1つのリストを含む。各生イベントが受け取られたとき、イベント・リダ600は、その生イベントのイベント・タイプを引き出し、そして次にそのイベント・タイプの関係するルールのリストを引き出すことにより、どの相関ルール620がその生イベントを処理すべきか判断する。そのどの相関ルール620がその生イベントを処理すべき1組の相関ルール620を判断したら、プロセスは図7のステップ45に戻る。

【0158】

次に図14を参照する。この図は、現在の生イベントが付け加えられた現在未熟な相関イベントが、対応するルール620の成熟基準を満たすすなわち満足するか否かを判断する図7のルーチン770のための、コンピュータにより実現されたプロセスを示す。ここに説明されるプロセスは、一般的なものではなく、1例としてのイベント・タイプ、攻撃されたホストからの攻撃(AFAH)、のためのものである。しかしながら、この処理の説明および先に行った例としての相関イベント・タイプの説明が与えられたので、この分野の専門家には、説明された例示のためのイベント・タイプの各々の発生を認識するために同じような処理がどのように使用できるか明らかなはずである。上に説明したように、各ル

ール620は、オブジェクト指向・アーキテクチャでは、1つのルール・オブジェクトとして実現することができる。先に説明された図13のステップ1315の処理からこの分野の専門家に明らかであろうように、単一の生イベントが複数の関連ルール・オブジェクトによって処理されてよい。

【0159】

図7または図14には示されていないが、図7のステップ745から780の処理（図14に説明されているルーチン770の処理を含む）は、この例示のためのAFAH関連イベントの場合には、現在処理されている生イベントのために、2回実行することもできる。1つの例としての実施の形態においては、生イベントは、外来する攻撃と見なされたときそれが完全な攻撃である場合にのみ一回処理され、また外出攻撃と見なされたときはいずれでも処理される。生イベントがステップ745から780によって外来する攻撃と見なされたときは、ステップ745は、関連イベント・キャッシュ665から対応するAFAH関連イベントを引き出そうと試みるときに、その生イベントの宛先インターネット・プロトコル・アドレスをルックアップ・キーとして使う（ステップ745の処理の解説において前に説明されているように）。

【0160】

その生イベントがステップ745から780によって外出攻撃と見なされたときは、ステップ745は、関連イベント・キャッシュ665から対応するAFAH関連イベントを引き出そうと試みるときに、その生イベントの送信元インターネットをルックアップ・キーとして使う（ステップ745の処理の解説において前に説明されたように）。この生イベントの「2重処理」は、フュージョン・エンジン22によって処理されることができる他の関連イベントに比べて、例としてのAFAH関連イベントの特有の面である。前に説明された代表例の関連イベント・タイプのすべてに対して、代表例の関連イベント・qタイプの説明に基づいてこの分野の専門家に明らかであろうように、ステップ745から780の処理は1回だけ行われる。

【0161】

再び図14を参照すると、ステップ1410はルーチン770の最初のステップである。このステップにおいて、処理されている現在の生イベントのためのクラシファイヤ615のイベント・タイプ・オブジェクトが、生イベント追跡インデックス645に追加される。また、処理されている現在の生イベント・オブジェクトに対応する関連イベント・オブジェクトが、メモリ管理リスト640に追加されるか（それが図7のステップ750において生成されたばかりの新しい関連イベントであるとき）、または現在の生イベントのタイムスタンプが現在の関連イベントと関係づけられる全ての生イベントのうちで最近のタイムスタンプを持つときはメモリ管理リストの新しい位置に移される。

【0162】

さらに、現在の関連イベント・オブジェクトは、生イベントと関連づけられた生イベント追跡インデックス645に追加される。イベント・タイプ・オブジェクトと関連イベント・オブジェクトは、それらが、後で、生イベントがメモリから消去される場合にメモリ管理処理によって通知され、それによりそれらがそれら自身のその生イベントに対する参照を消去することができるように、生イベント追跡インデックス645に追加される。現在の関連イベントも、メモリ資源が少なくなったときに、もっとも古いイベント（その中の幾つかは未熟の関連イベントであってもよい）がフュージョン・エンジン22から削除されることができるよう、メモリ管理リスト640に追加される。

【0163】

判断ステップ1415において、生イベントが外来攻撃と見なされているかどうか判断される。このステップは、現在のAFAH関連イベントが含まれる現在の生イベントが外来する攻撃または出ていく攻撃のどちらであるか識別する。問い合わせで判断するステップ1415が否定のときは、その生イベントは外出攻撃と見なされ、ステップ1425に向かって「ノー」枝が辿られる。問い合わせで判断するステップ1415が肯定のときは、その生イベントは外来する攻撃と見なされ、ステップ1420に向かって「イエス」枝が辿ら

10

20

30

40

50

れる。

【0164】

判断ステップ1420において、外来攻撃と見なされている生イベントが完全性攻撃でありかつ現在の相関イベントの外出攻撃リスト中の少なくとも1つのイベントよりも先に発生しているか否かが判断される。処理されつつある生イベントは、それがステップの処理中に相関イベントの外来攻撃リストに追加されたことから、完全性攻撃であることが知られる。代表例の相関イベント・タイプの前に示したリストに含まれている攻撃されたホストからの攻撃の説明中に示されているように、あるホストに対する完全性攻撃が見られ、その後にはそのホストから発出する秘密性攻撃、完全性攻撃、または利用可能性攻撃が続いたときは、AFAHイベントが生成される。図13の解説に示されるように、コントロール655は、システムの初期化中に相関ルール620をロードするときに、そのルールをそのルールにとって関心のあるイベント・カテゴリ中に含まれるイベント・タイプの全てと関係づける。

【0165】

AFAHルールの場合には、関心のあるイベント・カテゴリは秘密性攻撃、完全性攻撃、および利用可能性攻撃である。AFAHルールはしたがって生イベント分類データベース635によってこれらの3つのカテゴリの1つに属すると決められた全てのイベント・タイプと関係づけられる。したがって、そのイベント・タイプがこれらの3つのカテゴリの1つに属する生イベントはどれも、処理のためにルーチン770に送られることができる。AFAHイベントの定義は外来攻撃が完全性攻撃であることを要求し、そしてルーチン770に送られる生イベントの中にはそうではなく秘密性攻撃または利用可能性攻撃もあり得るので、この判断ステップ1420は外来攻撃と見なされている生イベントが完全性攻撃であることを確かめなければならない。

【0166】

フュージョン・エンジン22が複数のディテクタによって発生させられた生イベントを受け取ることができ、そのため生イベントが非日時順に受け取られ得る（すなわち、後のタイムスタンプを持つ生イベントがそれより前のタイムスタンプを持つ生イベントより先に受信され得る）可能性があるという事実から、さらに1つの問題が生じる。この理由のために、ルーチン770は生イベントが日時順に受け取られるであろうと見なすことはできず、したがってこの判断ステップ1420は現在の生イベントが現在の相関イベントの外出攻撃のリスト中の少なくとも1つのイベントよりも先に発生したか否かを判断する。問い合わせて判断するステップ1420が否定のときは、現在の相関イベントは未熟と見なされ、図7のルーチン785に向かって「ノー」枝が後方向に辿られる。問い合わせて判断するステップ1420が肯定のときは、現在の相関イベントは成熟していると見なされ、ステップ1427に向かって「イエス」枝が辿られる。

【0167】

判断ステップ1425において、外出攻撃と見なされている生イベントが現在の相関イベントの外来攻撃のリスト中の少なくとも1つのイベントよりも後に発生したか否かが判断される。判断ステップ1420と異なり、現在の生イベントが特定のカテゴリに属するか否かを判断することは必要でない、なぜなら（ステップ1420の解説において説明されているように）ルーチン770に送られる全ての生イベントは秘密性攻撃、完全性攻撃、または利用可能性攻撃のどれかであり、したがって外出攻撃としてどれかのAFAHイベントに含まれるための基準を満たすであろうからである。問い合わせて判断するステップ1425が否定のときは、現在の相関イベントは未熟と見なされ、図7のルーチン785に向かって「ノー」枝が後方向に辿られる。問い合わせて判断するステップ1425が肯定のときは、現在の相関イベントは成熟していると見なされ、ステップ1427に向かって「イエス」枝が辿られる。

【0168】

判断ステップ1427において、相関イベントの最も早い外来攻撃よりも前に発生した現在の相関イベント中の外出攻撃はみな外出攻撃のリストから削除される。これは、AFA

10

20

30

40

50

H 相関イベントの定義が、成熟 A F A H 相関イベントに含まれる各外出攻撃は少なくとも 1 つの外来攻撃によって先行されなければならないことを要求するために行われる。

【0169】

判断ステップ 1430 において、その相関イベントが、メモリ管理機構によって消去されないように、メモリ管理リスト 640 から削除される。このようにして、削除される相関イベントはフュージョン・エンジン 22 から消去されないであろう。なぜならその相関イベントは今や成熟していると考えられるからである。

【0170】

判断ステップ 1435 において、イベント・リーダ 600 によって読まれるイベント・ソースがイベント・データベース 26 またはイベント・ログ・ファイル 610 のどちらかであるときは、相関イベントの更新時刻は最近の生イベントのタイムスタンプに設定されることができる。この場合にはフュージョン・エンジン 22 はパッチ・モードで動作している。その代わりに、イベント・リーダ 600 によって読まれるイベント・ソースがイベント・コレクタ 24 のときは、相関イベントの更新時刻は、フュージョン・エンジン 22 が実行されているシステムの現在時刻に設定されることができる。この場合にはフュージョン・エンジン 22 はリアルタイム・モードで動作している。

【0171】

ステップ 1440 において、この相関イベントは成熟イベント相関リスト 650 に追加される。ステップ 145 において、2 つまたはそれ以上の生イベントを含む相関イベントが、その相関イベントの 1 つの内部パラメータを設定することにより、成熟であると標識される。プロセッサは次に図 7 のステップ 775 に戻る。1 つの代表としての実施の形態では、各相関イベントは、生イベントの優先順位パラメータと同じような優先順位を付けられてもよい。

【0172】

図 5D、5E、5F に示される生イベント I I のための代表例としてのルール処理

下記は、図 5D、5E および 5F に示されるような生イベント I I のための、攻撃を受けたホストからの攻撃の相関ルール 620 によって実行されるであろう処理である。この解説は、生イベント I および I I が両方とも完全性攻撃タイプであり、したがって A F A H イベントの定義による外来攻撃としての条件に適合していること、生イベント I は生イベント I I より先に発生していること、そして生イベント I I は生イベント I I I よりも先に発生していることを仮定している。

【0173】

再び図 7 を参照すると、ステップ 745 において、生イベント I I が外来攻撃であると見なされているとき、その宛先インターネット・プロトコル・アドレス (3. 3. 3. 3) が、相関イベント・キャッシュ 665 から A F A H 相関イベントを引き出すための探索キーとして使われるであろう。この場合において生イベントは日時順に受け取られ、したがって生イベント I I I はフュージョン・エンジンによってまだ処理されていないと仮定すると、攻撃されたインターネット・プロトコル・アドレス 3. 3. 3. 3 によって指し示される A F A H 相関イベントは存在せず、それゆえ判断ステップ 745 の「ノー」枝が辿られてステップ 750 で相関イベント 513 が生成されるであろう。ステップ 755 において、相関イベント 513 は相関イベント・キャッシュ 665 に記憶されるであろう。ステップ 760 において、生イベント I I が、それへの参照情報を相関イベント 513 の外来攻撃リスト中に記憶することにより、相関イベント 513 と関係づけられるであろう。ステップ 765 において、相関イベント 513 がまだ成熟していないと判断されて、ステップ 770 に向かって「ノー」枝が辿られるであろう。

【0174】

次に図 14 を参照すると、判断ステップ 1415 において、生イベント I I は外来攻撃と見なされているので、「イエス」枝が辿られるであろう。判断ステップ 1420 では、新しく生成された相関イベント 513 の外出攻撃リストには生イベントが無いので「ノー」枝が辿られるであろう。この処理を要約すると、外来攻撃と見なされているときは、生イ

10

20

30

40

50

イベント11は、新しく生成されたしかしまだ成熟していない相関イベント513に加えられる。

【0175】

図7を参照すると、ステップ745において、生イベント11が外出攻撃と見なされるときは、その送信元インターネット・プロトコル・アドレス(2.2.2.2)が、相関イベント・キャッシュ665からAFAH相関イベントを引き出すための探索キーとして使われるであろう。この場合において生イベントは日時順に受け取られ、したがって生イベント11はフュージョン・エンジンによってすでに処理されたと仮定すると、AFAH相関イベント511が、攻撃を受けたインターネット・プロトコル・アドレスで索引がつけられて、すでに相関イベント・キャッシュ665中に存在するであろう。したがって判断ステップ745の「イエス」枝が辿られてステップ760に進むであろう。ステップ760において、相関イベント513は相関イベント・キャッシュ665に記憶されるであろう。ステップ760において、生イベント11が、それへの参照情報を相関イベント511の外出攻撃リスト中に記憶することにより、相関イベント511と関係づけられるであろう。ステップ765において、相関イベント511がまだ成熟していないと判断されて、ステップ770に向かって「ノー」枝が辿られるであろう。

【0176】

次に図14を参照すると、判断ステップ1415において、生イベント11は外出攻撃と見なされているので、「ノー」枝が辿られるであろう。判断ステップ1425において、相関イベント511の外来攻撃リストはすでに生イベント11を含み、かつ生イベント11のタイムスタンプは生イベント11のタイムスタンプよりも後なので、「イエス」枝が辿られるであろう。この時点で、相関イベント511は成熟している判断され、そしてステップ1427から1445が辿られて新しく成熟した相関イベント511が処理されるであろう。この処理を要約すると、外出攻撃と見なされているときは、生イベント11は既存の相関イベント511に加えられ、相関イベント511はその結果成熟する。

【0177】

上に説明された例示のためのルール処理における判断ステップ1425を実行するために、最初に生成された生イベント1と2番目に生成された生イベント11のそれぞれのタイムスタンプが比較される。しかしながら、これらの生イベントは異なるディテクタから発し得るので、各生イベントに与えられるタイムスタンプにはいくらかのずれがあり得ることは注意されるべきである。すなわち、2番目に生成される生イベント11は1番目に生成される生イベント1の後に発生するはずであるが、生イベントを発生するディテクタの内部クロックの起り得るずれのために、1番目に生成される生イベント1が2番目に生成される生イベント11よりも後のタイムスタンプを持つ可能性があることは予見できる。

【0178】

換言すれば、隣接する侵入探知システム中の各ディテクタの間の内部クロックが同期させられていないこともあり得る。このような起り得る状況に対処するために、3状態比較を行うことができる。すなわち、フュージョン・エンジン22およびより具体的にはルール620は、第1の生イベントより先に来たかどうか判断を行うことができるように、多少の同期ずれがあるかもしれないという可能性を考慮に入れることができる。より具体的には、異なるディテクタによって生成された2つの生イベントのタイムスタンプを比較してそれらのイベントの1つが他方より先に起きた(または後に起きた)かどうかを決定するとき、比較の結果はイエス、ノー、またはかもしれない、であり得る。「かもしれない」という結果は、2つのイベントのタイムスタンプが、2つのディテクタの同期のずれに関する不確実性がどちらのイベントが先に起こったかを判断することを不可能にするほど近いときに発生する。

【0179】

フュージョン・エンジン22は、「かもしれない」という結果を「イエス」として(1つの構成において)または「ノー」として(別の構成)扱うように構成することができる。

好ましい実施の形態では、フュージョン・エンジン22は、相関イベントの成熟性の基準が満たされるであろう可能性を最大にするために（それらの基準が満たされているかもしれない可能性があるように見えるときはいつでも成熟相関イベントが生成されるであろうように）「かもしれない」を「イエス」として扱う。フュージョン・エンジン22が同一のディテクタによって生成された2つのイベントのタイムスタンプを比較するときは、同期の効果をいさい無視してそれらの2つのイベントのタイムスタンプの間の簡単な2値比較を行うことができる。

【0180】

成熟相関イベントがタイムアウトしたかどうかを判断するための例示のためのコンピュータで実現されたプロセス

次に図15を参照して、この図はどれかの成熟相関イベントが発生しなかったか否かを判断するルーチン785のためのコンピュータで実現されたプロセスを示す。ステップ1510はルーチン785の最初のステップであり、このステップにおいて現在の処理時刻が成熟イベント・リスト650に記憶されている相関イベントの更新時刻（相関イベントの更新時刻は図14のステップ1435に説明されているようにして設定される）と比較される。この比較の目的のためには、現在の処理時刻の定義はフュージョン・エンジン22が動作しているモードに依る。フュージョン・エンジン22がバッチ・モードで動作している（すなわち、そこからイベントが読まれつつあるイベント・ソースがイベント・データベース26またはイベント・ログ・ファイル610のどちらからである）ときは、現在の処理時刻はそのイベント・ソースから読まれた最新のイベントのタイムスタンプである。そうではなく、フュージョン・エンジンがバッチ・モードで動作している（すなわち、そこからイベントが読まれつつあるイベント・ソースがイベント・データベース26またはイベント・ログ・ファイル610のどちらからである）ときは、現在の処理時刻はそのイベント・ソースから読まれた最新のイベントのタイムスタンプである。そうではなく、フュージョン・エンジン22がリアル・モードで動作している（すなわち、そこからイベントが読まれつつあるイベント・ソースがイベント・コントローラ24である）ときは、現在の処理時刻はフュージョン・エンジン22が動いているシステムの現在時刻である。

【0181】

判断ステップ1515において、現在の処理時刻と各相関イベントの更新時刻の差が予め決められた閾値を越えたか否かを判断される。換言すれば、成熟イベント・リスト150中に含まれている成熟相関イベントが古くなったすなわち新鮮でなくなった、すなわちかなりの時間の間これらの相関イベントのためのコンピュータの活動または生イベントが発生しなかったかどうか判断される。問い合わせして判断するステップ1515が肯定のときは、図7のステップ790向かって「ノー」枝が後方向に辿られる。問い合わせして判断するステップ1515が否定のときは、図7のステップ795向かって「ノー」枝が後方向に辿られる。

【0182】

上の記載は本発明の説明のための実施の形態についてだけ説明していること、および請求の範囲により確定されている本発明の趣旨と範囲から逸脱することなくそれにおいて多数の変更が可能であることは理解されるべきである。

【図面の簡単な説明】

【図1】

図1は、本発明のための例示のための動作環境を提供するネットワークに接続されたパーソナル・コンピュータのブロック線図である。

【図2】

図2は、本発明のための例示のためのネットワーク・アーキテクチャを示す機能ブロック線図である。

【図3】

図3は、本発明のための例示のためのソフトウェア・アーキテクチャを示す機能ブロック線図である。

10

20

30

40

50

【図 4】

図 4 は、本発明のための例示のためのソフトウェアおよびハードウェア・アーキテクチャを示す機能ブロック線図である。

【図 5 A】

図 5 A は、コンピュータのインシデント源についての情報をフュージョン・エンジンに接続されたイベント・コレクタに供給するセキュリティ情報のデータ・ソースを示す機能ブロック線図である。

【図 5 B】

図 5 B は、侵入探知システム中のディテクタによって生成される生イベント中に存在することができるデータのタイプを示す説明図である。

【図 5 C】

図 5 C は、フュージョン・エンジンの CoBRA プロセッサによって処理された例示のための生イベントを示す説明図である。

【図 5 D】

図 5 D は、攻撃されたホスト・コンピュータからの攻撃というセキュリティに対する脅威を示す機能ブロック線図である。

【図 5 E】

図 5 E は、図 5 D に基づく例示のための相関イベントの実際にあり得るデータを示す説明図である。

【図 5 F】

図 5 F は、図 5 D に基づくもう 1 つの例示のための相関イベントの実際にあり得るデータを示す説明図である。

【図 6】

図 6 は、図 2 に示されているフュージョン・エンジンの幾つかの構成要素を示す機能ブロック線図である。

【図 7】

図 7 は、1 つまたはそれ以上のデータ・ソースから集められたセキュリティ情報を管理するための方法の例示のための実施の形態を示す論理フロー図である。

【図 8】

図 8 は、図 6 に図示された図 7 および 9-15 に関して解説されているいろいろなソフトウェア・コンポーネントの間の情報の交換を示すデータ・フロー図である。

【図 9】

図 9 は、リアルタイムの生イベントをイベント・タイプのリスト中の 1 つまたはそれ以上のカテゴリに配属させるための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

【図 10】

図 10 は、各リアルタイム生イベントにコンテキスト・パラメータを指定するための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

【図 11】

図 11 は、各リアルタイム生イベントの優先順位を調整するための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

【図 12】

図 12 は、各リアルタイム生イベントの優先順位を調整するための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

【図 13】

図 13 は、リアルタイムの生イベントのデータに対応するルールに送るための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

【図 14】

図 14 は、相関イベントが成熟しているか否かを判断するための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

10

20

30

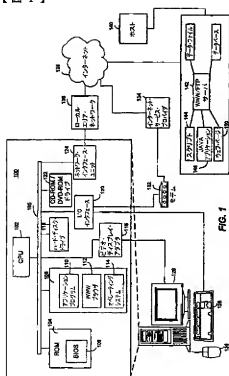
40

50

【図 15】

図 15 は、成熟関連イベントが発生を停止しているか否かを判断するための図 7 の例示のためのサブプロセスまたはルーチンを示す論理フロー図である。

【図 1】



【図 2】

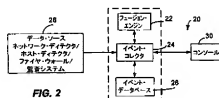


FIG. 2

【図 3】

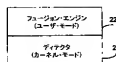


FIG. 3

【図5F】

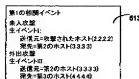


FIG. 5F

【図6】

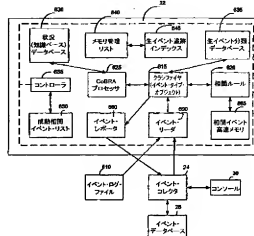


FIG. 6

【図7】

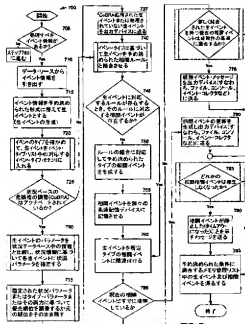


FIG. 7

【図8】

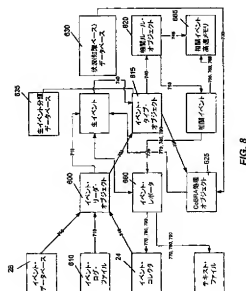


FIG. 8

【图 9】



FIG. 9

【☒ 10】

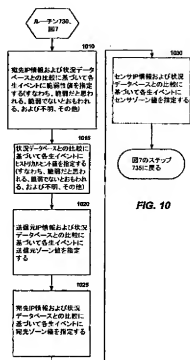


FIG. 10

【☒ 1 1】

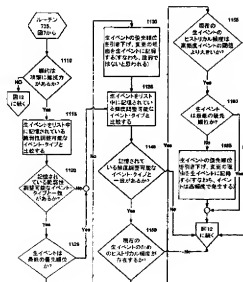


FIG. 11

【 1 2 】

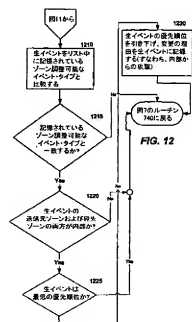


FIG. 12

【図13】



FIG. 13

【図14】

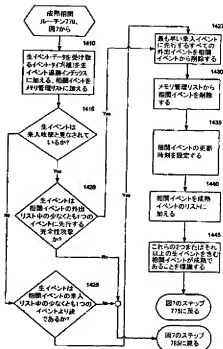


FIG. 14

【図15】

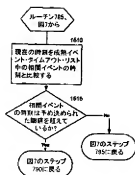
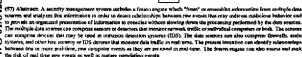


FIG. 15

WO 01/84285 A2

(10) International Publication Number
WO 01/84285 A2



WO 01/84285

PCT/US01/43799

METHOD AND SYSTEM FOR MANAGING COMPUTER SECURITY
INFORMATION

PRIORITY AND RELATED APPLICATIONS

- 5 The present application claims priority to provisional patent application entitled, "Intrusion Detection Fusion Systems of a Network Security System," filed on April 28, 2000 and assigned U.S. Application Serial Number 60/200,316. The present application is also related to non-provisional application entitled, "System and Method for Managing Security Events on a Network," (Attorney Docket No. 05456-105005) filed on April 27, 2001 and assigned U.S. Application Serial Number _____.
- 10

TECHNICAL FIELD

- The present invention relates to computer systems and the security of such systems. More particularly, the present invention relates to a method and system for tracking individual security events according to risk and timing or identifying relationships between two or more security events that may occur on or within a computer system. The invention also identifies relationships in other security related information.
- 15
- 20

BACKGROUND OF THE INVENTION

- The nature of a distributed network, such as the internet, makes it vulnerable to attack. The internet was designed to allow for the freest possible exchange of information, data, and files. However, this free exchange of information carries a price: many users will try to attack the networks and computers connected to the internet; many users will also try to invade other users' privacy and attempt to crack databases of sensitive information or intercept information as it travels across internet routes.
- 25

- To detect or prevent such computer attacks, intrusion detection systems (IDS) and software programs that gather information and make changes to security configurations of network computers have been developed. However,
- 30

WO 01/04285

PCT/US98/03799

these conventional intrusion detection systems can typically have many problems and drawbacks. Conventional intrusion detection systems typically comprise hardware that is dedicated to intrusion detection on networks. Other intrusion detection systems can simply comprise programs running on a host computer.

5 The problems and drawbacks of many conventional intrusion detection systems can be attributed to at least two parameters that are part of any detection design. The first parameter is the speed in which a detector of an intrusion detection system must run in order to be transparent to the data or communication that flows through the detector. Detectors that typically run on dedicated personal
10 computers must be able to handle constantly increasing loads of informative traffic, as network speeds increase from 100 megabits per second to gigabit per second speed and beyond. Because of these high speeds, a detector of an intrusion detection system cannot perform complex analysis of the information that flows through the detector for obvious reasons. That is, if a detector were to
15 perform complex analysis of the information flowing through it, then such analysis would fail to keep up with the flow of information that passes through the detector.

A second key parameter that is part of any detection design is typically the volume of information that may pass through a detector. Because of the high
20 speed at which information passes through a detector, a detector must be able to analyze high volumes of data packets.

In light of current network speeds and the corresponding volume of information that is generated as a result of the network speeds, many detectors of conventional intrusion detection systems can provide very limited protection
25 against complex and more sophisticated computer attacks. This limited protection can manifest itself when many false positives are generated by an intrusion detection system. In other words, many conventional intrusion detection systems may generate false alarms based on communications between computers that do not comprise any threat or attacks.

WO 01/84285

PCT/US99/13779

In addition to false alarms, conventional intrusion detection systems are typically not equipped to handle complex analysis because of the limitations on current processing speeds. For example, many conventional intrusion detection systems cannot execute central processing unit-intensive checks such as the well-known L0pht Crack. The L0pht Crack decodes can use cryptographic challenge-response data from Windows (SMB) connections to crack passwords in use on a network. The conventional method for executing L0pht Crack is to obtain packets using a packet-capturing tool and then crack the passwords offline. Conventional intrusion detection system typically cannot employ the L0pht Crack method in any real-time analysis.

Another obstacle of conventional intrusion detection systems is that most intrusion detection systems have very limited or short term memory capacity. In other words, long histories of data streams are seldom kept by the detectors in conventional intrusion detection systems.

Another problem of conventional intrusion detection systems is that the detectors of such systems typically only watch or observe a single environment. For example, detectors usually observe only parts of networks. Conventional detectors typically have a limited scope of awareness since they are designed to observe only portions of a network instead of the entire network as a whole. Because conventional detectors typically monitor only portions of a network, they are unable to track more sophisticated computer attacks such as distributed attacks.

In addition to the inability to track more sophisticated computer attacks, many conventional intrusion detection systems do not permit active probing of an attacker or the target of a computer attack. Active probing typically involves making a determination to see whether a computer attack has had an effect on its target. Further, probing can also comprise methods for discovering additional information about an attacker. However, as mentioned above, most intrusion detection systems do not permit active probing since such probing could reveal

WD 0184285

PCT/IB03/01779

the location of the detector. And if the location of a detector is revealed, it sometimes may also become a target for a computer attack.

Accordingly, there is a need in the art for a method and system for managing security information for a network. That is, there is a need in the art to log, investigate, respond to, and track computer security incidents that may occur in a network computer system. There is also a need in the art to determine whether security within a network or over a network has been compromised or if an incident is just some odd behavior that should be disregarded by an intrusion detection system. Another need exists in the art for a method and system that can monitor and analyze security information from multiple data sources so that rather complex and sophisticated computer attacks can be identified, stopped, or prevented. A further need exists in the art for a method and system for managing security information in real-time.

Another need exists in the art for a method and system for managing security information such that it can be determined if one or more real-time computer events are related to each other and if they are a part of a larger scheme or sophisticated attack. An additional need exists in the art for a method and system for managing security information where multiple computer events can be correlated together if the computer events are part of a larger scheme or attack.

Another need exists in the art for a method and system for managing security information where computer events that are detected can be prioritized so that attention can be focused on those computer events which could cause the most damage to a network or individual computers. Similarly, another need exists in the art for a method and system for managing security information that enables rapid response to existing computer attacks in addition to prevention of the additional computer attacks which may spin off from or be generated from a single computer attack. A further need exists in the art for a method and system for managing security information such that real-time computer events can be classified and ranked according to their respective priorities in the context of the environment in which the event occurred.

WO 01/84385

PCT/IB01/01799

SUMMARY OF THE INVENTION

The present invention can solve the aforementioned problems by providing a computer security management system that can log, investigate, respond to, and track computer security incidents that can occur in a networked computer system. The invention can track suspicious computer activity or actual computer security threats. Actual security threats can include, but are not limited to, integrity attacks, confidentiality attacks, denial of service attacks, multi-stage attacks, or other similar attacks on computers or computer networks. The invention typically refers to suspicious computer activity descriptions obtained from data sources as real-time raw events and actual computer security threats as mature correlation events. The invention can comprise a method and system for managing security information collected from one or more data sources. More specifically, the present invention can comprise a fusion engine which "fuses" or assembles information from multiple data sources and analyzes this information in order to detect relationships between raw events that may indicate malicious behavior and to provide an organized presentation of information to one or more consoles without slowing down the processing performed by the data sources.

The multiple data sources can comprise sensors or detectors that monitor network traffic or individual computers or both. The sensors can comprise devices that may be referred to as intrusion detection systems (IDS). Because the present invention can be separate from IDS devices, it permits the IDS devices to operate efficiently and at high speeds when real-time processing of high volumes of data traffic is essential.

The data sources can also comprise firewalls and other fire security or IDS devices. Further, the data sources can comprise any devices that may or may not provide real-time information, such as audit systems, that provide additional environmental information about a network or computer or internet. For example, one data source could comprise a database. The database may include a raw event classification database that organizes categories of different types of raw events. Another database can comprise a context or knowledge database that includes

WO 01/04285

PCT/JP01/1759

network context information, such as host vulnerability status, historical computer event frequency values, and network zone definitions.

From the multiple data sources, the fusion engine of the present invention can correlate and identify real-time, raw computer events. That is, unlike the conventional art which usually processes computer events after some period of time, the present invention can identify relationships between one or more real-time, raw computer events as they are received in real-time. Real-time raw computer events or raw events may comprise any computer activity that may be tracked by an intrusion detection system as a possible attack on a computer or a plurality of computers. Raw events can be generated by detectors of intrusion detection systems. Each raw event may comprise various parameters that may include, but are not limited to the following: source internet protocol address of the computer activity, destination internet protocol address of the computer activity, priority status assigned by the detector, a vulnerability status assigned by the detector, a time stamp, and an event type parameter.

The fusion engine can determine if one or more real-time raw events are related to each other and if they are part of a larger scheme or computer attack. Real-time raw events that are related to each other and that may indicate that a computer attack may be occurring are referred to by the fusion engine as a mature occurrence event. A correlation event can comprise one or more raw events. However, a correlation event does not mean an actual security threat or attack has been detected. Correlation events typically store related raw events and usually indicate that a security event or computer attack has occurred when the correlation event is deemed to be mature. In order to be deemed mature, a correlation event must satisfy the criteria or algorithm of a corresponding correlation rule. Therefore, it is possible to track numerous correlation events that may comprise one or more raw events that have not yet been identified as being a mature correlation event or actual computer security threat or computer attack.

The fusion engine can also assess and rank the risk of real-time raw events as well as mature correlation events based on information about the environment or context in which the event occurred. The fusion engine can display this risk and

WO 01/84285

PCT/US96/03779

rank information as messages on a console. The fusion engine can generate and send updates related to mature correlation events to a console. Further, the fusion engine can determine and indicate when a mature correlation event has stopped occurring.

- 5 In order to assess risks and determine ranks of real-time raw events, the fusion engine can utilize the aforementioned raw event classification database and the knowledge database. The raw event classification database can permit the fusion engine to classify raw computer events while the knowledge database can permit the fusion engine to rank and evaluate the risk of a raw computer event based upon the context of the raw computer event. The raw event classification database can comprise one or more tables of security information. That is, the raw event classification database can comprise tables that include information that can categorize raw events based on their impact on the target host (confidentiality, integrity, or availability), their scope (network, host, or service), and the method they employ (jackknifing, IDS evasion or detection evasion, etc.).
- 10 15 The context of the raw computer event can be determined by comparing parameters of the raw event with context parameters in a context or knowledge database, such as the aforementioned event vulnerability statuses, historical computer event frequency values, and event definitions.

- 20 To determine if one or more raw computer events are part of or form a mature correlation event, the fusion engine can apply one or more rules that can be triggered based upon how the fusion engine classifies a raw computer event. In other words, the rules applied by the fusion engine can be automated and applied to raw computer events according to the classification (identification of the type or kind) of the raw events.
- 25

- In addition to determining whether raw computer events are part of or form a mature correlation event or actual security threat, the fusion engine can also manage its high speed memory processes very efficiently. For example, the fusion engine can employ memory management techniques that erase raw events, immature, and mature correlation events that have either exceeded a predetermined time period or that have met predetermined conditions or both. The
- 30

W/O 0184185

PCT/US01/03799

high speed memory resources can comprise RAM containing data that is categorized according to the classifications of the raw events and mature correlation events.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a network personal computer that provides the exemplary operating environment for the present invention.

Fig. 2 is a functional block diagram illustrating exemplary network architecture for the present invention.

Fig. 3 is a functional block diagram illustrating an exemplary software architecture for the present invention.

Fig. 4 is a functional block diagram illustrating exemplary software and hardware architecture for the present invention.

Fig. 5A is a functional block diagram illustrating security information data sources feeding information about a computer incident source to an event collector that is connected to a fusion engine.

Fig. 5B is a diagram illustrating the type of data that may be present in a raw event generated by a detector in an intrusion detection system.

Fig. 5C is a diagram illustrating an exemplary raw event that has been processed by the CoBFA processor of the fusion engine.

Fig. 5D is a functional block diagram illustrating an exemplary attack from attacked host computer security threat.

Fig. 5E is a diagram illustrating the possible data of an exemplary correlation event that is based on Fig. 5D.

Fig. 5F is a diagram illustrating the possible data of another exemplary correlation event that is based on Fig. 5D.

Fig. 6 is a functional block diagram illustrating some components of the fusion engine illustrated in Fig. 2.

Fig. 7 is a logic flow diagram illustrating an exemplary embodiment of a method for managing security information collected from one or more data sources.

WO 01/84285

PCT/US01/17799

Fig. 8 is a data flow diagram illustrating the exchange of information between various software components that are illustrated in Fig. 6 and discussed with reference to Figs. 7, and 9-15.

Fig. 9 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for assigning real-time raw events to one or more categories in an event type list.

Fig. 10 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for assigning context parameters to each real-time raw event.

Fig. 11 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for adjusting the priority status of each real-time raw event.

Fig. 12 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for adjusting the priority status of each real-time raw event.

Fig. 13 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for forwarding real-time raw event data to corresponding rules.

Fig. 14 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for determining whether a correlation event is mature.

Fig. 15 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for determining whether a mature correlation event has stopped occurring.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

The present invention may be embodied in program modules that run in a distributed computing environment. The present invention can comprise a computer security management system that can log, investigate, respond, and track computer security incidents that can occur in a network computer system. The present invention can comprise a fusion engine which "fuses" or assembles

WO 01/64285

PCT/RS98/13799

information from multiple data sources and analyze this information in order to provide an organized, and sometimes ranked, presentation of information to one or more consoles. The fusion engine can classify raw real-time computer events while also ranking the real-time computer events based upon comparisons with one or more databases.

Illustrative Operating Environment

Although the illustrative embodiment will be generally described in the context of six program modules running on a personal computer and a server, those skilled in the art will recognize that the present invention may be implemented in conjunction with operating system programs or with other types of program modules for other types of computers. Furthermore, those skilled in the art will recognize that the present invention may be implemented in either a stand-alone or in a distributed computing environment or both. In a distributed computing environment, program modules may be physically located in different local and remote memory storage devices. Execution of the program modules may occur locally in a stand-alone manner or remotely in a client server manner. Examples of such distributed computing environments include local area networks and the Internet.

The detailed description that follows is represented largely in terms of processes and symbolic representations of operations by conventional computer components, including a processing unit (a processor), memory storage devices, connected display devices, and input devices. Furthermore, these processes and operations may utilize conventional computer components in a heterogeneous distributed computing environment, including remote file servers, computer servers, and memory storage devices. Each of these conventional distributed computing components is accessible by the processor via a communication network.

The processes and operations performed by the computer include the manipulation of signals by a processor and the maintenance of these signals within data structures resident in one or more memory storage devices. For the

WO 01/04285

PCT/US98/03799

5 purposes of this discussion, a process is generally conceived to be a sequence of computer-executed steps leading to a desired result. These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is convenient for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

15 It should also be understood that manipulations within the computer are often referred to in terms such as creating, adding, calculating, comparing, moving, receiving, determining, identifying, populating, loading, unloading, etc. that are often associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the computer.

20 In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the program modules constructed in accordance with the teachings described herein. Similarly, it may prove advantageous to construct a specialized apparatus to perform the method steps described herein by way of dedicated computer systems in a specific network architecture with hard-wired logic or programs stored in nonvolatile memory, such as read-only memory.

25 Referring now to the drawings, in which like numerals represent like elements throughout the several Figures, aspects of the present invention and the illustrative operating environment will be described.

WO 01/41285

PCT/US98/13799

Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Referring now to Fig. 1, an illustrative environment for implementing the invention includes a conventional personal computer 100, including a processing unit 102, a system memory, including read only memory (ROM) 104 and random access memory (RAM) 106, and a system bus 108 that couples the system memory to the processing unit 102. The read only memory (ROM) 104 includes a basic input/output system 106 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 100, such as during start-up. The personal computer 100 further includes a hard disk drive 118 and an optical disk drive 122, e.g., for reading a CD-ROM disk or DVD disk, or to read from or write to other optical media. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer 100. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM or DVD-ROM disk, it should be appreciated by those skilled in the art that other types of media are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the illustrative operating environment.

A number of program modules may be stored in the drives and RAM 106, including an operating system 114 and one or more application programs 110, such as a program for browsing the world-wide-web, such as WWW browser 112. Such program modules may be stored on hard disk drive 118 and loaded into RAM 106 either partially or fully for execution.

A user may enter commands and information into the personal computer 100 through a keyboard 128 and pointing device, such as a mouse 130. Other control input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 102 through an input/output interface 120 that is coupled to the system bus, but may be connected by other interfaces, such as a game port, universal serial bus, or firewire port. A display monitor 126 or other

WO 01/84385

PCT/US00/12799

type of display device is also connected to the system bus 105 via an interface, such as a video display adapter 116. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers. The personal computer 100 may be capable of displaying a graphical user interface on monitor 126.

The personal computer 100 may operate in a networked environment using logical connections to one or more remote computers, such as a host computer 140. The host computer 140 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer 100. The LAN 136 may be further connected to an Internet service provider 134 ("ISP") for access to the Internet 138. In this manner, WWW browser 112 may connect to host computer 140 through LAN 136, ISP 134, and the Internet 138. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 100 is connected to the LAN 136 through a network interface unit 124. When used in a WAN networking environment, the personal computer 100 typically includes a modem 132 or other means for establishing communications through the Internet service provider 134 to the Internet. The modem 132, which may be internal or external, is connected to the system bus 105 via the input/output interface 120. It will be appreciated that the network connections shown are illustrative and other means of establishing a communications link between the computers may be used.

The operating system 114 generally controls the operation of the previously discussed personal computer 100, including input/output operations. In the illustrative operating environment, the invention is used in conjunction with Microsoft Corporation's "Windows NT" operating system and a WWW browser 112. However, it should be understood that the invention can be implemented for use in other operating systems, such as Microsoft Corporation's "WINDOWS 3.1," "WINDOWS 95," "WINDOWS 98" and "WINDOWS 2000" operating systems, IBM Corporation's "OS/2" and "AIX" operating systems, SunSoft's

WO 01/54285

PCT/US98/13799

"SOLARIS" operating system used in workstations manufactured by Sun Microsystems, and the operating systems used in "MACINTOSH" computers manufactured by Apple Computer, Inc. Likewise, the invention may be implemented for use with other WWW browsers known to those skilled in the art.

5 Host computer 140 is also connected to the Internet 138, and may contain components similar to those contained in personal computer 100 described above. Additionally, host computer 140 may execute an application program for receiving requests for WWW pages, and for serving such pages to the requester, such as WWW server 142. WWW server 142 may receive requests for WWW
10 pages 150 or other documents from WWW browser 112. In response to these requests, WWW server 142 may transmit WWW pages 150 comprising hyper-text markup language ("HTML") or other markup language files, such as eXtensible Markup Language (XML), to WWW browser 112. Likewise, WWW server 142 may also transmit requested data files 148, such as graphical images or text
15 information, to WWW browser 112. WWW server 142 may also execute scripts 144, such as CGI, PERL, ASP, or JSP (Java Server Pages) scripts, to dynamically produce WWW pages 150 for transmission to WWW browser 112. WWW server 142 may also transmit scripts 144, such as a script written in JavaScript, to WWW browser 112 for execution.

20 Similarly, WWW server 142 may transmit programs written in the Java programming language, developed by Sun Microsystems, Inc., to WWW browser 112 for execution. The WWW server 142 could comprise a UNIX platform running Apache or Netscape webserver. Alternatively, the WWW server 142 could comprise an Internet Information Server (IIS). The present invention is not
25 limited to these documented examples. Other web server environments are not beyond the scope of the present invention.

As will be described in more detail below, aspects of the present invention may be embodied in application programs executed by host computer 142, such as scripts 144, or may be embodied in application programs executed by computer
30 100, such as Java applications 146. Those skilled in the art will also appreciate

WO 01/04265

PCRU061/3799

that aspects of the invention may also be embodied in a stand-alone application program.

Exemplary Computer Architecture

5 Referring now to Figure 2, the computer architecture for one exemplary embodiment of the present invention will be described. Figure 2 illustrates the System 20 for managing security information collected from one or more data sources. The security system 20 can comprise a fusion engine 22 that is linked to an event collector 24. The event collector 24 can comprise an event sink or device that can organize events received from multiple data sources in a logical manner. Further details of the event collector 24 are described in a related application entitled, "System and Method for Managing Security Events on a Network," (Attorney Docket No. 05456-105005) filed on April 27, 2001 and assigned U.S. Application Serial Number _____, the contents of which is hereby incorporated by reference.

15 The security management system 20 can further comprise an event database 26 that is also linked to the event collector 24. The security management system can also comprise data sources 28 that are linked to the event collector 24 and a console 30 which is also linked to event collector 24. Information from the databases are typically loaded into fusion engine 22 that comprises high-speed memory devices such as random access memory (RAM) since comparisons between raw events and the databases must be performed in a very rapid and in a very efficient manner. Most memory resources used in the fusion engine 22 comprise high-speed memory devices such as RAM (sometimes referred to as "caches" hereinbelow). However, other memory resources are not beyond the scope of the present invention. The memory resources of the fusion engine 22 should be designed to handle high volumes of information with increased speed.

25 The one or more data sources 28 can comprise many different hardware and software devices. For example, a data source 28 can comprise a network detector or a host detector. Similarly, a data source 28 could also comprise a

WO 01/84285

PCT/US98/03799

firewall or an audit system. The present invention is not limited to the types of data sources illustrated. The function of a data source 28 is to provide the event collector 24 with various types of information as it may relate to the network, host, or single computer being monitored by the security management system 26. Other like data sources 28 are not beyond the scope of the present invention. One data source 28 can comprise a host detector which monitors network traffic in the form of data packets. Another data source 28 could comprise observations made by users who are monitoring any network or computer activity.

The one or more data sources 28 forward their information to the event collector 24. The event collector 24 may comprise one or more program modules designed to store and collect the data received from the one or more data sources 28. The event collector 24 can arrange the data and store it in the event database 26. The event collector 24 also forwards any information received from the data sources 28 to the fusion engine 22. The detectors 28 of intrusion detection systems scan raw network traffic or host system events for predefined patterns. Once the detectors identify these predefined patterns of information, the detectors generate a raw event which is then sent to the event collector and later to the fusion engine 22. The fusion engine assembles or "fuses" the raw events or information received from the event collector 24. In other words, the fusion engine 22 organizes and analyzes the information received from the one or more data sources 28 in order to provide an organized presentation of information by correlating (identifying relationships between) raw computer events that are related to each other.

Once the fusion engine 22 determines that two or more events are related to each other (to form a "correlation" event), the fusion engine 22 generates messages and forwards these messages to the event collector 24. The event collector 24, in turn, forwards the messages generated by the fusion engine 22 to a console 30.

Console 30 may comprise a program module that runs on a separate personal computer. The fusion engine 22 may comprise one or more program modules running on a personal computer. The fusion engine 22, the event

WO 01/04385

PCT/US98/13799

collector 24, and the event database 26 have been circumscribed by a box 32 to demonstrate that each of these software components can reside on a single computer. However, the present invention is not limited to this configuration.

And therefore, the fusion engine 22, the event collector 24, and the event database 26 could also reside on separate computer devices. Other combinations of the software components illustrated could be implemented. That is, the fusion engine 22 and event collector 24 could reside on one hardware device while the event database 26 resides on another hardware device. Conversely, the event collector 24 and event database 26 could reside on one hardware device while the fusion engine 22 resides on another hardware device. Those skilled in the art will appreciate that disclosed software architecture is not limited to the architecture illustrated in the drawings.

Referring now to Figure 3, a functional block diagram illustrating another exemplary software architecture for the present invention is illustrated. In Figure 3, the fusion engine program module 22 and a data source such as a detector module 28 could reside in a single machine. That is, the high speed I/O functions of the detector 28 could reside only the kernel of a computer while the fusion engine 22 could reside in the user mode part of the computer. In this way, the additional processing of the fusion engine 22 would not slow down the high speed intrusion detection system functions performed by the detector 24.

Referring now to Figure 4, this Figure illustrates another functional block diagram of exemplary software and hardware architectures for the present invention. In this one exemplary embodiment, the data source 28 comprising a detector could be implemented in a hardware device such as a detector board or a detector chip so that the high speed intrusion detection system functions could be performed. In this exemplary embodiment, the fusion engine 22 could simply reside as a program module in software. Figure 4 demonstrates that the data sources 28 that require access to high speed data streams can be separated from the fusion engine 22 such that network processing speeds can be achieved without significant interpretation or delay on both.

WO 01/84265

PCT/US98/02799

Referring now to Figure 5A, this Figure illustrates a functional block diagram of the security information data sources 25 feeding information about a computer incident source 500 to the event collector 24 which is also connected to the fusion engine 22. Figure 5A further illustrates a network 510 that may comprise numerous data sources 25, user work stations 520, a server 530 that is a target for a computer incident source 500, an internal router 540, and the server 550. The network 510 is connected to the internet 590 by an external router 560 and by a firewall 28. The firewall 28 can comprise a bastion host or similar device. The firewall 28 can also be connected to an internal screening router 540 that may examine all packets of data travelling to and from the internal screening router 540. The user work stations 520 can be stand-alone personal computers that access the server 530, 550.

The computer incident source 500 can be a computer or a network of computers that originate an attack against the network 510 and more specifically, the server 530 (attached host). The computer incident source 500 can be connected to the server 560 of a local area network. Alternatively, instead of a server 560, the computer incident source 500 can be connected to a dial-in internet service provider (ISP) or any computer connected to the internet. The server 560 or ISP (or other computer connected to the internet) can then be connected to a router 570. A router 570 provides access to a distributed computer network such as the Internet 590.

While the computer incident source 500 can be located outside of the network 510, it is possible for the computer incident source 500 to be located within the network 510. That is, a computer incident source 500 could be a user workstation 520 located within the network 510. For example, in case of a disgruntled employee within a company, a user workstation 520 could be used as the computer incident source 500 when the employee decides to interfere or hamper the operation of a network 510 or one or more other workstations 520 within the network 510.

WG 0184285

PCT/US81/1799

Each of the data sources 28 has a data line illustrated by dashed lines that feed into the event collector 24. The dashed data lines could comprise actual physical data lines, however, these data lines are more for illustrative purposes to demonstrate that each of the data sources is equally linked to the event collector 24. Also, the event collector 24 could reside within the network 518 so that it would not be vulnerable to a direct attack from a computer incident source 580. The placement of event collector 24 within Figure 5 illustrates the collection function of the event collector 24. Figure 5 illustrates fundamental concepts of a system for managing security information rather than the actual physical architecture that would be implemented to support such a system.

Exemplary Data Processed by Fusion Engine

Referring now to Figure 5B, this diagram illustrates an exemplary raw event 585 that is generated by a detector of an intrusion detection system. The raw event 585 may comprise a source Internet protocol address 515; a destination Internet protocol address 525; a priority status 535; a detector assigned vulnerability status 545, an event type parameter 555; and a time stamp 565. As will be discussed in further detail below, the priority status 535 assigned by detectors of an intrusion detection system are typically very conservative in nature. That is, since detectors must process information very quickly, they are unable to run complex algorithms or tests to ascertain the risk of certain computer raw events. Therefore, the priority status 535 of many raw events generated by detectors will be very high relative to an actual priority of a raw event.

Referring now to Figure 5C, this Figure is a diagram illustrating a CoBRA-(Control Based Risk Adjustment) processed raw event. The CoBRA-processed raw event 592 typically contains all of the previously detector assigned parameters of the raw event and in addition the CoBRA-processed parameters that may comprise any one of the following: a CoBRA-assigned vulnerability value 594; a CoBRA-assigned historical frequency value 596; a CoBRA-assigned source zone value 598; a CoBRA-assigned destination zone value 510; a

WG 0184285

PCT/US98/03799

CoBRA-assigned sensor score value 512; a CoBRA-assigned original priority status 514; and a priority change reason 516 text string comprising a reason why the priority of the raw event was adjusted (if adjusted). These CoBRA-assigned values will be discussed below in further detail with respect to Figures 11 and

5 Figure 12.

Exemplary Raw and Correlation Events Processed by Fusion Engines

Referring now to Figure 5D, this Figure is a functional block diagram illustrating an exemplary Attack From Attacked Host (AFAH) computer security threat. Figure 5D illustrates a computer incident source 503 with an Internet protocol address of 1.1.1.1 sending an attack to host (attacked host) 505 that has an Internet protocol address of 2.2.2.2. The attack between the computer incident source 503 and the attacked host 505 may be characterized as a raw computer event I. After being attacked, the attacked host 505 then sends another attack to a second host 507, having an Internet protocol address of 3.3.3.3. The attack between the attacked host 505 and the second host 507 may be characterized as a second raw event II. The second host 507 generates an attack on a third host 509, having an Internet protocol address of 4.4.4.4. The attack between the second host 507 and third host 509 may be characterized as a third raw event III.

After processing the raw events I, II and III, the fusion engine 22 may identify the relationships between respective raw events. Therefore, after processing the raw events illustrated in Figure 5B, the fusion engine may generate a mature correlation event 511 that corresponds to the first and second raw events I and II. Further, the fusion engine 22 may further generate a second mature correlation event 513 that identifies a relationship between the second and third raw events II and III. Further details of the processing performed by the fusion engine 22 to generate the first and second mature correlation events 511 and 513 will be discussed below in further detail with respect to Figure 7 and Figure 14.

Referring now to Figure 5E, this Figure is a diagram illustrating the possible data of an exemplary correlation event that is based on Figure 15. The

WO 01/41285

PCT/JP00/03799

correlation event 511 illustrated in Figure 5B may comprise two sets of lists. The first list may identify inbound attacks relative to the attacked host 505 and outbound attacks relative to the attacked host 505. Further details of the first exemplary correlation event 511 will be discussed in further detail below with respect to Figure 7 and Figure 14.

Referring now to Figure 5F, this Figure is a diagram illustrating the possible data of the second correlation event 513 illustrated in Figure 15. The second correlation event 513 may also comprise two lists: one list identifying inbound attacks relative to the second host 507 and a second list identifying outbound attacks relative to the second host 507. Further details of the second mature correlation event 513 will be discussed below with respect to Figure 7 and Figure 14.

The exemplary attack from attacked host computer security threat illustrated by Figures 5D through 5F is just but one example of the possible computer security threats that can be analyzed with the fusion engine 22. As discussed above and below, other types of computer security threats are not beyond the scope of the present invention. In one exemplary embodiment, the fusion engine may track at least twenty different types of possible correlation events. Those skilled in the art will appreciate the present invention is not limited to the exemplary correlation events illustrated in Figure 5D and that fewer or more correlation events can be utilized by the present invention without departing from the scope and spirit thereof.

Exemplary Software Components of Fusion Engine

Figure 6 is a function block diagram illustrating some components of the fusion engine 22 that is illustrated in Figure 2. Basically, Figure 6 illustrates some of the numerous software components that make up the software architecture for the fusion engine 22.

The present invention includes a computer program which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the

WO 01/04285

PCT/US98/13799

invention in computer programming, and the invention should not be construed as limited to any one set of computer program instructions. Further, a skilled programmer would be able to write such a computer program to implement the disclosed invention without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions is not considered necessary for an adequate understanding how to make and use the invention. The inventive functionality of the claimed computer program will be explained in more detail in the following description in conjunction with the remaining Figures illustrating the program flow.

In one exemplary embodiment, the fusion engine 23 can be implemented with object-oriented programming. Therefore, some of the software components illustrated in Figure 6 can have both data and code associated with a respective software object. However, the general functionality of each software object will be generally described such that a skilled programmer will be able to write a computer program to implement the disclosed functionality of the software object.

The fusion engine 23 may comprise several software components. In the exemplary embodiment illustrated in Figure 6, the fusion engine 23 may comprise an event reader 600 that receives raw computer event information from the event collector 24. The event reader 600 is operably linked to the classifier 615. The classifier 615 organizes the raw event information received from the event reader 600. In other words, the classifier 615 categorizes the raw event information by separating the raw event information according to an event type property that is included in each raw event. The event type property of each raw event is typically generated by a detector in an intrusion detection system.

The classifier 615 can be responsible for forwarding raw event information to the CuBRA processor 625 and one or more correlation rules 620. The one or more correlation rules 620 may comprise algorithms for testing and determining whether a security incident may be occurring. The correlation rules track raw event information that is received from the classifier and stores the raw event

WG 01/04/05

PCT/US01/17799

information in correlation event high speed memory 645. The correlation event high speed memory 645 may comprise random access memory (RAM) for storing information. However, the present invention is not limited to RAM type memory. Other high speed memory devices are not beyond the scope of the present invention. The classifier 615 can be established based upon a raw event classification database 635. The classifier 615 can be generated upon initialization of the fusion engine 22 when event classification data is read from the raw event classification database 635 into the classifier 615.

The CoBRA processor 625 may comprise algorithms or software components for the context based risk adjustment of raw computer events. The CoBRA processor 625 can adjust the priority values of raw computer events by comparing a raw computer event against data contained within a context or knowledge base database 636. The priority status of raw events is typically established by detectors of intrusion detection systems before forwarding the raw event data to the fusion engine 22. After processing raw computer events, the fusion engine 22 can inform the event collector 24 whether a security event is occurring. The fusion engine 22 typically formats and sends one or more correlation events to the event collector via the event reporter 668. As noted above, a correlation event may comprise one or more computer events that are related to each other as determined by the fusion engine 22.

The fusion engine 22 may further comprise memory management devices that can conserve memory resources for the fusion engine 22. For example, in one exemplary embodiment, the fusion engine 22 may comprise a memory management list 640, a raw event tracking index 645 and a mature event list 650. The memory management list 640 is typically linked to the raw event tracking index 645. Further details of the functionality with respect to the memory management list 640, raw event tracking index 645, and the mature event list 650 will be discussed below in the brief process description of the software components illustrated in Figure 6.

30

WO 01/04285

PCT/US98/03799

Exemplary Object-Oriented Architecture for Figure 6

One of the software components of the fusion engine 22 that can be implemented as a software object, in one exemplary embodiment, is the event reader 600. The event reader 600 can receive raw computer events from either the event collector 34 or an event log file 610. The event log file 610 can comprise files having columns separated values (CSV) formats that store computer event data from an intrusion detection system. The event reader 600 typically reads in raw computer events or raw events which can be any computer activity that may be tracked by an intrusion detection system, as a possible attack on a computer or a plurality of computers. The event reader typically creates raw event data objects (not shown) that are processed by other software components on a fusion engine 22.

In one exemplary embodiment, the event reader 600 can be linked to a classifier 615 which may categorize one or more event type objects. The classifier 615 receives the raw event objects that are generated by the event reader 600. A classifier 615 associates each raw event object with a corresponding event type object that has been established for a specific event type parameter 555. In other words, the classifier assigns raw event objects to event type objects according to the type of raw event. It is noted that each raw event received by the event reader 600 has been assigned a type or categorization based upon the intrusion detection system that generated the raw event.

One function of the classifier 615 is to categorize or classify each of the raw events and then forward the raw event objects to specific correlation rules 620 based upon their type. The correlation rules 620 can also take form of software objects that receive the raw event objects from the classifier 615.

The classifier 615 can also forward the raw event object to the Context Based Risk Adjustment (CoBRA) processor 625. The CoBRA processor is a risk assessment mechanism that can adjust priority parameters of raw event objects. The CoBRA processor 625 accesses a context or knowledge base database 630 in order to perform its context based risk adjustment for each received raw event

WO 01/04285

PCVRI01U3799

object. Basically the CHSRA processor determines the risk of a raw computer event by assessing the event type parameter 555 in combination with environmental factors such as the destination internet protocol address of an attack in addition to the source of the attack.

- 5 The context or knowledge base database 638 can include vulnerability stations assigned to machines within a network, historical event frequency values, and network zone definitions. The vulnerability stations can be results of vulnerability scans performed by devices outside of the fusion engine 22 that determine the strength or resistance of a network or single computer to an attack.
- 10 The historical event frequency value can comprise signatures or data relating to computer attacks that occur over very long periods of time. The network zone definitions can comprise values assigned to parts of a network based upon the amount and type of information that may be accessible in certain parts of a network. For example, it is useful to distinguish internal, external, and demilitarized zones as will be discussed below.
- 15

- The fusion engine 22 can further comprise a raw event classification database 635 that can be responsible for establishing the different event type objects which form the classifier 615. The raw event classification database 635 can comprise one or more tables of security information. The tables can include
- 20 information relating to the type parameter 555 of a raw event assigned by detectors. The raw event classification database 635 can categorize raw events based on their impact on the target host (confidentiality, integrity, or availability), their scope (network, host, or service), and the method they employ (bufferoverflow, cloning, etc.). Confidentiality events can be those events that indicate an attacker is attempting to obtain information from or about a host. Integrity events can be those events that indicate an attacker is attempting to alter data on a host, possibly to gain unauthorized access.

- Availability events can be those events that indicate an attacker is attempting to cause a denial of service, such as by causing a host to crash. In
- 30 addition to the above general criteria, specialized criteria useful in recognizing

WO 01/04285

PCT/US98/03799

particular correlation events can serve as a basis for identifying events. For example, events that confirm the success of a denial of service attempt can be grouped into a category used by a correlation rule 620 that identifies denial of service attacks that are believed to have succeeded. However, the raw event classification database 635 is not limited to these categories or parameters. Other categories and parameters which further define raw events are not beyond the scope of the present invention.

The fusion engine 22 can further comprise a memory management list 640, a raw event tracking index 645, and a mature event list 650. The memory management list 640 enables the fusion engine 22 to manage its memory resources by eliminating or deleting the oldest raw events when memory resources exceed a predetermined threshold (i.e., when memory resources are running low). The memory management list 640 can be implemented as a software object that deletes raw events that are considered oldest when memory resources run low. Related to the memory management list 640 is the raw event tracking index 645 which can also be implemented as another software object. A raw event tracking index 645 can identify which software objects may contain a particular raw event object. Thus, the raw event tracking index identifies those software objects that may be storing a raw event that has now become old and should be deleted from the fusion engine 22.

Related to the memory management list 640 and raw event tracking index 645 is the mature correlation event list 650 which tracks those raw events that have been identified as a pattern of activity or as actual computer threat that should not be removed from the memory management list 640. In other words, the mature correlation event list identifies the raw events which should not be deleted from the fusion engine 22 since these events are deemed to be part of mature correlation events or actual computer security threats.

The fusion engine 22 may further comprise a controller 655 that may be responsible for the data flow between respective software objects. In other words,

WO 01/04285

PCT/US01/3709

the controller 655 can be implemented as a high level software object which controls the data flow between lower level software objects.

The fusion engine 22 may further include the event sequencer 660 that can also be implemented as a software object in the exemplary and prefabricated object-oriented programming environment. The event sequencer 660 can be a software object that receives mature correlation events which are forwarded to the event collector 24. Mature correlation events can comprise one or more raw events that are associated together because the one or more raw events may pose an actual computer security threat.

10

Computer-Implemented Process for Managing Security Information

Referring now to Figure 7, this Figure illustrates an exemplary logic flow diagram of a computer-implemented process for managing security information collected from one or more data sources. More specifically, the logic flow diagram illustrated in Figure 7 illustrates a computer-implemented process for fusing or assembling security information received from multiple data sources and analyzing the security information in order to provide an organized presentation of information to one or more consoles. The logic flow described in Figure 7 is the core logic of the top-level processing loop of the fusion engine 22, and as such is executed repeatedly so long as the fusion engine 22 is operating.

20

It is noted that the logic flow diagram illustrated in Figure 7 illustrates a process that covers after initialization of several of the software components illustrated in Figure 6. That is, in the exemplary object-oriented programming architecture of the present invention, several of the software components or software objects that are required to perform the steps illustrated in Figure 7 are initialized or created prior to the process described by Figure 7. Therefore, one of ordinary skill in the art recognizes that several steps pertaining to initialization of the software objects illustrated in Figure 6 are not illustrated. For example, as noted above, the software component or software object comprising the classifier 615 is established after initialization of the fusion engine 22.

30

WG 8184285

PCT/D581/13799

During initialization of the fusion engine 22, the classifier 615 is built by reading information from the raw event classification database 635. The classifier 615 may comprise a comprehensive list of event type objects corresponding to the types of raw events that can be processed by the fusion engine 22, and a distinct event type object list for each event category defined in the raw event classification database 635. Each distinct event type list can contain the subset of the comprehensive event type list that constitutes the set of raw event types defined by the raw event classification database 635 as belonging to one category. While the initialization of the various software components illustrated in Figure 6 are not described with specificity, a skilled programmer would be able to write such computer programs to implement the disclosed invention without difficulty based upon the following flow charts and associated description of the software architecture in the current application.

Certain steps in the processes described below must naturally precede others for the present invention to function as described. However, the present invention is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present invention. That is, it is recognized that some steps may be performed before or after other steps without departing from the scope and spirit of the present invention.

Referring back to Figure 7, this figure provides an overview of the core logic of the top-level processing loop of the entire computer security management process where step 705 is the first step of the process 700. In decision step 705, it is determined whether there are any raw events to be processed by the fusion engine 22. As described above, raw events may comprise computer events reported from detectors or intrusion detection systems. Raw computer events identified by intrusion detection systems may include various parameters. For example, in one exemplary embodiment, each raw event may comprise a source internet protocol address, a destination internet protocol address, the type of computer event being reported, a priority status, a vulnerability status, and a time stamp.

WO 01/64285

PCT/US98/03799

If the inquiry to decision step 705 is negative, then the "no" branch is followed in which the process proceeds to step 785. If the inquiry to decision step 705 is positive, then the "yes" branch is followed to step 710 in which the raw computer events or event information is retrieved from a data source. The data source may comprise at least one of the event database 26, an event log file 610, or the event collector 24 as illustrated in Figure 8.

Referring briefly to Figure 8, this Figure is a data flow diagram illustrating the exchange of information between various software components that are illustrated in Figure 6. This data flow diagram of Figure 8 parallels the steps described in Figure 7. For example, step 710 for retrieving event information from data sources is illustrated in Figure 8 where the event reader object 600 in the exemplary object-oriented software architecture reads in event information. References to Figure 8 will be made throughout the detail description of Figures 7.

Referring back to Figure 7, after step 710 and in step 715, the event information or raw events are arranged and assigned a predefined format referred to as raw events. In other words, in the exemplary object-oriented programming environment, the event reader object 600 can create software objects for each raw event as it is received from one of the data sources such as the event database 26, the event collector 24, and the event log file 610. The event reader 600 generates the raw event objects in response to commands received from the controller 655. In other words, the controller 655 requests the event reader 600 to retrieve raw events from each of the data sources.

After step 715, in routine 720, the event type from each raw event is ascertained and each raw event is then assigned to a corresponding event type object in an event type list. In other words, in the exemplary object-oriented software architecture, each raw event object that is created by the event reader 600 is sent to a corresponding event type object that is present within the classifier 615. Further details of routine 720 will be discussed with reference to Figure 9.

WO 01/04285

PCT/US98/03799

Next, in decision step 725, it is determined whether the context based risk adjustment processor (CoBRA) 625 is activated. In other words, a user may elect to not adjust any of the priority status information that is present in each raw event. As noted above, each raw event generated by a detector in an intrusion detection system typically contains parameters drawn to the priority of the event. That is, the detectors of intrusion detection systems assign relative values to computer events to measure the risk or potential damage that could be associated with a raw event. For example, a distributed attack against a network could be assigned a higher priority status relative to a computer attack against a single machine or computer.

If the inquiry decision step 725 is negative, then the "no" branch is followed to routine 740. If the inquiry decision step 725 is positive, then the "yes" branch is followed to routine 730 in which parameters of a raw event are compared with information in the context or knowledge base database 630. Also in this routine, context parameters are assigned for each raw event based upon the context information present in the context database 630. Referring briefly to Figure 8, the classifier 615 containing the event type objects forwards each raw event to the CoBRA processing object or CoBRA processor 625. In routine 730, the CoBRA processor 625 can assign context parameters that relate to the environment or surrounding conditions of a raw event.

Following routine 730, in routine 735, the priority status of each raw event can be adjusted or the original status can be left intact based upon the CoBRA assigned context parameters or detector assigned type parameters or both of the raw event. Basically routines 730 and 735 can encompass the exemplary algorithms and methods of the CoBRA processor 625. Further details of routines 730 and 735 will be discussed below with respect to Figures 10, 11, and 12.

Next, in step 737, the CoBRA processed raw event or unprocessed raw event can be sent to an output device, such as the event collector 24. The event collector 24 then typically stores the CoBRA processed raw event or unprocessed raw event in the event database 26 and then forwards the event to the console 30.

WO 01/84285

PCT/US00/3799

As will become apparent from the discussion below, the console 30 can be provided with unprocessed raw events, ColIRA processed raw events, and correlation events. All such events can be handled by the fusion engine 22 and forwarded by the event collector 24 so that they can be displayed to user. It is noted that when a raw event is received by the event collector 24 from a data source 28, the event collector first sends the raw event to the fusion engine 22. However, if after a predetermined amount of time, the fusion engine 22 does not respond, then the event collector 24 will store the event in the event database 26 and then forward the unprocessed (not handled by the fusion engine 22) raw event to the console 30.

In routine 740, the raw event is associated with correlation rules 620 based upon the event type assigned by a detector 28. In this routine, the classifier 615 containing the event type objects determines which correlation rule(s) 620 should process the raw event based upon the event type parameter 555. Further details of routine 740 will be discussed below with respect to Figure 13.

In decision step 745, if a rule corresponding with a raw event exists, then it is determined whether a correlation event exists that is related to the correlation rule. Note that although depicted as a single process flow in Figure 7, steps 745 through 780 are actually performed independently for each Correlation Rule 620 associated with the raw event. Basically, in decision step 745, the correlation rule object or correlation rule 620 determines if a correlation event object has been created for the current raw event being processed. As illustrated in Figure 8, the correlation rule objects or correlation rule 620 check the correlation event cache or correlation event high speed memory 665 to determine whether the correlation event for the current raw event being processed has been created. As noted above, the correlation event (or correlation event object in an object-oriented software architecture) can comprise a number of raw events that are grouped together to form a single higher level event.

For step 745, each Correlation Event has an anchor Internet protocol (IP) address that is used to locate the Correlation Event in the Correlation Event type's

WO 01/64285

PCT/JP01/01779

area within the correlation event cache 665. The anchor IP address will be the source IP address or destination IP address of one or more of the Raw Events within the Correlation Event, depending on the Correlation Event type. For example, the anchor IP address of the Attack From Attacked Host event is the IP address of the attacked host. This is the destination IP address of inbound attacks, and the source IP address of outbound attacks. The Correlation Rule for the Attack From Attacked Host event uses the destination IP address of an inbound Raw Event as the Correlation Event lookup key when attempting to retrieve the Correlation Event for which the Raw Event would be an inbound attack. The AFAll Correlation Rule uses the source IP address of the Raw Event as the Correlation Event lookup key when attempting to retrieve the Correlation Event for which the Raw Event would be an outbound attack.

If the inquiry to decision step 745 is positive, then the "yes" branch is followed to step 746. If the inquiry to decision step 745 is negative, then the "no" branch is followed to step 750 in which a correlation event of the predetermined type associated with the current correlation rule is created. That is, in the exemplary object-oriented software architecture, at this point in processing of a raw event's associated correlation rules 620, one or more correlation event objects can be created.

Next, in step 755, the correlation events are stored in the high speed memory devices 665. The high speed memory devices in one exemplary embodiment can comprise random access memory (RAM). However, other high speed memory devices are not beyond the scope of the present invention. Because of current network processing speeds and the corresponding volumes of information, it is necessary to use high speed memory devices like RAM so that rapid processing of raw information can be obtained.

In step 780, the raw event is associated with the corresponding correlation event (which was either just created in step 750, or retrieved from the correlation event cache 665 in step 745) based upon the type of raw event. In other words, in this step in the exemplary object-oriented software architecture, each correlation

WO 01/04285

PCT/US98/13799

event object stores the raw event based upon its type. In addition to associating the raw event with the correlation event, the raw event tracking index 645 is updated to indicate that the raw event is associated with the correlation event.

Next in decision step 765, it is determined whether the current correlation event being processed is already mature. Typically, to be mature, a correlation event can contain two or more raw events that meet maturity criteria defined for that specific type of correlation event. The maturity criteria for each correlation event type are defined to identify the conditions under which the occurrence of two or more raw events indicates that a likely security incident is occurring. In step 765, the correlation event is being examined to determine if it had already been deemed mature as a result of processing of an earlier raw event.

If the inquiry in decision step 765 is positive, then the "yes" branch is followed to step 786. If the inquiry to decision step 765 is negative, then the "no" branch is followed to routine 776. In routine 776, it is determined whether the current correlation event with the newly associated raw event being processed meets or fulfills the maturity criteria set forth in one or more of the correlation rules 626. In routine 776, each of the rules that correspond to the type of raw event being processed determines whether the current raw event and any other raw events listed in the current correlation event together satisfy the maturity criteria as set forth in the rule. The present invention can include any number of rules that correspond to the type precursor of a given raw event.

In one exemplary embodiment, the fusion engine 32 can employ numerous correlation rules 626. The correlation rules can use the event categories defined in the Raw Event Classification Database 635 as a basis for identifying event patterns that indicate either malicious or nonmalicious activity. Many of the correlation events and corresponding correlation rules can reveal the intent of an attacker. The set of correlation events detected by the present invention and corresponding correlation rules includes, but is not limited to, the following:

WO 01/64285

PCT/00/013797

- 1) Attack From Attacked Host. This event can be generated when an Integrity attack is seen against a host followed by a Confidentiality, Integrity, or Availability attack originating from that host.
- 2) Availability Attack Sweep (Multihost DoS Attack). This event can be generated when two or more different types of Availability attacks originating from the same source IP address are seen against multiple target IP addresses.
- 3) Confidentiality Attack Sweep (Multihost Information Gathering). This event can be generated when two or more types of Confidentiality attacks are seen originating from a single source IP address against multiple target IP addresses.
- 4) DoS Followed By Confirming Event. This event can be generated when an Availability attack is seen against a target IP address followed by another event indicating that the target is no longer behaving normally. Confirming events include events detected by a network-based sensor indicating the host is not reachable (for example, detection of ARP requests from other hosts for the target), and events detected on the target system itself by a host-based sensor indicating that system resources (such as memory) have become exhausted.
- 5) External Source Using Internal IP Address. This event can be generated when a network-based sensor that monitors an external network detects a duplicate internal IP address. The occurrence of this condition indicates that an external host is attempting to use the IP address of an internal host, a practice known as spoofing.
- 6) Integrity Attack Followed By Remote Login. This event can be generated when an Integrity attack is seen against a host followed by a remote login originating from that host.
- 7) Integrity Attack Followed By Start Of Service. This event can be generated when an Integrity attack is seen against a host followed by a report from a host-based sensor that a new service has been started on the host.

WO 01/84283

PCT/US00/13799

- 8) Internet Scanner Scan. This event can be generated when an IDS Internet Scanner scan is detected from a host. For a period following detection of the start of the scan, all other events originating from the same host are subsumed into the Internet Scanner Scan event. If the source IP address is configured as an approved scan source, the event can be treated as a nonmalicious event; otherwise it can be treated as a malicious event.
- 9) Probe Followed By Integrity Attack. This event can be generated when a Probe event is seen against a host followed by an Integrity attack against the host.
- 10) Integrity Attack Sweep (Polling For Victims). This event can be generated when two or more types of Integrity attacks are seen originating from a single source IP address against multiple target IP addresses.
- 11) Login From DoS-attacked Host. This event can be generated when a remote login is seen from a source IP address that is currently the target of an ongoing Availability attack. This combination of events can indicate that an attacker is masquerading as a particular host (the target of the Availability attack) in order to exploit network trust relationships to access other machines on the network.
- 12) Login Failure Of One User On Multiple Hosts. This event can be generated when login failures of the same user are reported by multiple network- or host-based sensors.
- 13) Suspicious Activity Followed By Availability Attack. This event can be generated when an event that involves a Cloaking method is reported, followed by an Availability attack. The term "cloaking" applies to any technique that attempts to conceal an attack from intrusion detection systems.
- 14) Suspicious Activity Followed By Integrity Attack. This event can be generated when an event that involves a Cloaking method is reported, followed by an Integrity attack. The term "cloaking" applies to any technique that attempts to conceal an attack from intrusion detection systems.

WO 01/94285

PCT/US00/13799

- 15 Suspicious Activity Followed By Integrity Attack: This event can be generated when an event that involves a *Cloning* method is reported, followed by an integrity attack. The term "cloning" applies to any technique that attempts to conceal an attack from intrusion detection systems.
- 16 Sustained Availability Attack (Focused DoS Attack): This event can be generated when two or more types of Availability attacks are seen from a single source IP address targeted at a single destination IP address.
- 17 Sustained Confidentiality Attack (Focused Information Gathering Attack): This event can be generated when two or more types of Confidentiality attacks are seen from a single source IP address targeted at a single destination IP address.
- 18 Sustained Integrity Attack (Focused Break-in Attempt): This event can be generated when two or more types of Integrity attacks are seen from a single source IP address targeted at a single destination IP address.
- 19 Web Scan: This event can be generated when multiple Web-related attacks targeted against a Web server are detected within a certain interval. By examining features of the Web-related attacks such as the sequence of URLs being probed, it can be possible to identify the use of specific Web scanning tools such as *Whisker*.

Additional rules may be employed without departing from the scope and spirit of the present invention. Further details of routine 770 will be disclosed in further detail below with respect to Figure 14. However, it is noted that routine 770 as illustrated in Figure 14 only covers the application of one rule. The exemplary rule illustrated in Figure 14 is the rule corresponding to the "Attack From Attacked Host" (AFAH) correlation event listed above. The attack from attacked host scenario will also be described in further detail below with respect to Figures 5D through 5F.

If the inquiry to decision routine 770 is negative, then the "no" branch is followed to decision routine 785. If the inquiry to decision routine 770 is positive,

WO 01/4285

PCT/US00/3799

then the "yes" branch is followed to step 775 in which a mature event message is generated and forwarded to an output device such as the event collector 24. In step 775, the event reporter 660 receives an indication that the correlation event is mature and then the event reporter 660 forwards this message to the event collector 24.

In step 780, a correlation event update notification is sent to the output device when a new event is added to a correlation event that is already mature. In this step, the event reporter forwards the correlation event update to event collector 24 which, in turn, updates the representation of the correlation event in the Event Database 26 and forwards this information to the console 30 where it may be viewed by a user. This allows the user to be notified when additional new events occur that are part of an ongoing security incident (i.e., a mature correlation event).

Next, in decision routine 785, it is determined whether any mature correlation events have stopped occurring. Further details of decision routine 785 will be discussed below with respect to Figure 15.

If the inquiry to decision routine 785 is negative, then the "no" branch is followed to step 795. If the inquiry to decision step 785 is positive, then the "yes" branch is followed to step 790 in which a message is sent indicating that a correlation event has stopped occurring. This message can be forwarded from the event reporter 660 to the event collector 24. In turn, the event collector 24 would update the representation of the now-concluded correlation event in the event database 26 and then forward this message to the console 30.

In step 795, the oldest raw events and immature correlation events in the memory management list may be erased. Because the fusion engine 22 has a limited amount of memory, it is necessary to keep the memory filled with raw events that are the most likely to become mature. The fusion engine has several memory usage monitoring devices such as the memory management list 640, the raw event tracking index 645, and the mature event list 650. In one exemplary embodiment, the memory usage monitoring devices of the fusion engine

WO 01/64285

PCT/US01/13799

determine how much memory is available and when the memory is close to being filled to capacity, the memory usage monitoring devices will erase the oldest existing stored raw events and transform correlation events in order to increase available memory. Raw events that are included within mature correlation events are removed from the memory management list 640 but are not erased. Whenever a raw event is deleted, the raw event tracking index 645 is used to locate any immature correlation events that contain the raw event, and the raw event is removed from those immature correlation events. When a raw event is removed from an immature correlation event and the immature correlation event then contains no other raw events, the immature correlation event is also erased.

Referring now to Figure 9, this Figure illustrates the computer-implemented process for routine 728 of Figure 7 which identifies the type of raw event and assigns each raw event to a corresponding event type object of the classifier 615. Routine 728 begins with step 918 where each raw event is matched with a corresponding event type in the classifier 615. Next, in step 915, the time stamp of each raw event is identified. In step 920, each raw event is added to the memory management list 640 based upon the time stamp identified in step 915. The entries in this list are typically maintained in order by timestamp to facilitate locating the oldest events during the memory cleanup processing described above. In step 925, each raw event is stored in the high speed memory device associated with its event type object as contained in the classifier 615. Next, in step 930, each event type object receiving a raw event is added to the raw event tracking index 645. That is, typically, each software component of the fusion engine registers itself with the raw event tracking index 645 upon receiving a raw event. In this way, when a raw event is determined to be deleted from the system, the raw event tracking list 645 can be used to identify the location of the raw event references that need to be erased. After step 930, the process returns back to decision step 725 of Figure 7.

Figure 10 illustrates the computer-implemented process for routine 730 of Figure 7 in which parameters of each raw event are compared with the content of knowledge base database 630. Also in this routine, additional parameters are

WO 01/04285

PCT/0091/1799

assigned to each raw event based upon this comparison with the context database 630. As noted above, the context database 630 can comprise environmental information that may be helpful in evaluating the importance of a raw event. For example, the context database 630 can comprise vulnerability information about machines or computers within a network, the relative location of a computer or detector based upon predetermined zones, and information relating to historical event frequency.

The vulnerability information of the context database 630 is usually derived from scans made across a network to determine the relative security risk that may be present at one or more machines that make up a network. A tool that analyzes historical raw event logs for the network being monitored by the fusion engine 22 typically derives the historical event frequency information of the context database 630. This tool typically calculates average event frequencies for groups of events sharing the same raw event type, source internet protocol address, and destination internet protocol address, though other approaches to grouping raw events for the purpose of calculating average event frequencies could be used and are within the scope of the present invention. The zone definitions of the context database 630 are usually derived by categorizing parts of a network as they relate to the entire network. For example, an internal zone and demilitarized zone (DMZ) may be defined such that the internal zone includes the internet protocol network addresses of the networks that should not be accessible from the Internet, and the DMZ zone includes the internet protocol network addresses of the networks that are accessible from the Internet. These zones would be defined as appropriate for the specific network being monitored by the fusion engine 22.

Routine 750 is typically performed by the CoBRA processor 625. The CoBRA processor 625 typically examines each raw event and compares it to the context database 630. More specifically, in step 1819 (the first step of Routine 750) a CoBRA vulnerability status 504 is assigned for each raw event based upon destination internet protocol address information and a comparison with the context database 630. In one exemplary embodiment, the vulnerability value

WO 01/84185

PCT/US98/03799

assigned can be any one of the following: believed vulnerable; believed not vulnerable; and unknown.

- Next, in step 1815, a historical frequency value 506 is assigned for each raw event based upon another comparison with the context database 636. This value can be a number of events per unit time, such as events per day, or a mathematically related value, such as an average time between events. The historical event frequency value typically indicates how frequently raw events of a particular type from a particular source machine to a particular destination machine are seen on the network being monitored by the fusion engine 22.
- 10 Historical frequency data is used by the fusion engine to aid in distinguishing events caused by normal non-malicious network activity from those caused by unusual and malicious activity.

- In step 1820, a source zone 508 value is assigned to each raw event based upon the source Internet protocol address of the raw event and a comparison with the context database 638. In step 1825, a destination zone 510 value is assigned to each raw event based upon the destination Internet protocol address of each raw event and a comparison with the context database 638.

- In step 1830, a sensor zone 512 value is assigned to each raw event based upon the sensor Internet protocol address and a comparison with the context database 630. The sensor zone value can comprise the Internet protocol address of the sensor or detector of an intrusion detection system that detected the suspicious computer activity and generated the raw event. After step 1830, the process returns to routine 735 of Figure 7.

- Referring now to Figure 11, this Figure illustrates the computer-implemented process for routine 735 of Figure 7, which can adjust the priority status or leave an original priority status of a raw event intact based upon the CoBRA-assigned context parameters or detector-assigned type parameters or both. Routine 735 is another core function of the CoBRA processes 625. This routine enables the fusion engine to rank raw events based upon their importance to a network or a computer being protected. In this way, the security

WO 01/04285

PCT/US91/03799

administrator can monitor computer security events more efficiently and effectively, since the important computer security events will have a higher ranking and priority relative to other lower-level security events.

The present invention can employ user-defined attributes for those events and parts of a network that are most important to a user. For example, the zone definitions that form part of the context database 630 can be supplied by the user. In one exemplary embodiment, an internal zone and a so-called demilitarized zone (DMZ) of the monitored network can be established by the user. Rather than being explicitly defined by the user, an external zone can be any IP address that doesn't fall within an explicitly defined zone or zones supplied by the user. The present invention is not limited to these types of zones and can include other types of zones such as a business partner zone, for example. Those skilled in the art will appreciate that the present invention can be designed to associate Internet protocol addresses with any number of zones defined by a user.

As noted above, each raw event comprises a priority status parameter 535 that was assigned to it by a detector within an intrusion detection system. In one exemplary embodiment, the priority status parameter can comprise any one of the following three values: 1, 2, or 3. The highest priority status value in the exemplary embodiment is typically the number 1. Meanwhile, the lower priority status in the exemplary embodiment is typically the value of 3. The mid-range priority value is typically the number 2. Adjustments of the priority status values for each raw event are necessary since the priority status values assigned by the detector typically are very conservative in nature. That is, raw events are typically the result of simple processing techniques that must occur at the detector level in order to maintain high network traffic speeds.

Therefore, the priority status values coming from the detector level of conventional intrusion detection systems typically are defined as appropriate for the worst-case scenario that could apply for each event type. For example, in the exemplary embodiment, if a given type of raw event could have an actual priority of 1, 2, or 3 depending on the circumstances that apply on a given network, the

WO 01/64285

PCT/US91/03799

detector would typically assign the worst-case priority (denoted in one (1) in the exemplary embodiment) to all events of this type. Whenever the CoBRA processor 625 modifies the value of the priority status 535, it does so only after storing the both the original detector-assigned priority status and the updated or CoBRA-adjusted priority parameter in priority status 535. That is, in the exemplary embodiment, after CoBRA processing and if priority is adjusted, the adjusted priority status 535 will contain two values: the original detector-assigned priority status and the CoBRA-adjusted priority status.

The fusion engine 23 permits the ranking of raw events based upon the environmental conditions or surroundings in which a raw event is generated. In this way, the security network administrator will only be presented with the computer security events that are most important to the network or computer being monitored. The present invention is not limited to the priority status scale illustrated. That is, the present invention is not limited to a priority scale that spans between 1 and 3 where one is designated as the highest priority. Other ranges or values and increments between values are not beyond the scope of the present invention. Those skilled in the art will appreciate that more complex scales can be generated to further reduce the possibility of ranking an unimportant raw event over an important raw event.

Routine 735 begins with step 1110, in which it is determined whether a target of a raw event is resistant to the computer attack. This determination is made based on the CoBRA vulnerability status 504 value of the raw event previously established by step 1010 of procedure 730 described in Figure 10. If the inquiry to decision step 1110 is negative, then the "no" branch is followed where the process continues to step 1210 of Figure 12. If the inquiry to decision step 1110 is positive, then the "yes" branch is followed to step 1115. In step 1115, the raw event is compared to vulnerability-adjustable event types stored in a list in the content database 630. These vulnerability-adjustable event types stored in the content database 630 are events identified by either a user or a system for which the assessment of a machine's vulnerability status is believed trustworthy

WO 01/04285

PCT/US00/1799

and for which therefore it is allowed to adjust priority based on vulnerability status information.

Alternatively, in another embodiment (not illustrated) the context database 630 can identify those raw event types for which a user or system does not believe the assessment of vulnerability status to be trustworthy, and the assessment of vulnerability status can be deemed trustworthy for all other event types. In this way, raw events that are not desired to be adjusted with respect to their priority status can be identified so that the ColERA process 625 will not reduce the priority of such raw events. In another alternative exemplary embodiment (not shown), the context database 630 can also contain both types of lists. That is, the context database 630 can comprise a list of raw event types that are permitted to have the priority status to be adjusted and a list containing raw event types that are not permitted to have the priority status adjusted. In this case a conflict resolution rule must also be established, so that if a particular event type appears in both lists, it is well-defined which entry will take precedence. Those skilled in the art will appreciate that other configurations of lists are not beyond the scope of the present invention.

Next, in decision step 1120, it is determined whether a match exists with the stored vulnerability-adjustable events of the context database 630. If the inquiry to decision step 1120 is negative, then the "no" branch is followed to step 1135. If the inquiry to decision step 1120 is positive, then the "yes" branch is followed to decision step 1125.

In decision step 1125, it is determined whether the current raw event being processed is at its lowest priority status. In other words, if the current raw event being processed has an exemplary priority status value of 3, then it is recognized that its priority cannot further be adjusted. Therefore, if the inquiry to decision step 1125 is positive, then the "yes" branch is followed to step 1135. If the inquiry to decision step 1125 is negative, then the "no" branch is followed to step 1130, in which the priority status 535 of the current raw event is reduced and the reason for the change in priority status 535 is recorded in the raw event. For

WO 01/04285

PCT/US01/03799

example, in the exemplary embodiment, if a raw event has an original priority status value of a 1, and if the CoBRA processor 635 determines that the raw event is not believed to be vulnerable, then it will adjust the original priority status value of 1 to a lower value such as the value of 2 (the mid-range priority status value).

- 5 The reason for changing the priority status value is recorded in the priority change reason 516 parameter of the raw event so that it can be determined at the console 30 why a particular raw event was assigned a reduced priority. In one exemplary embodiment, the reason for changing priority status of a raw event can comprise a text string.

- 10 In step 1135, each raw event is compared to frequency-adjustable event types stored in a list in the context database 638. Similar to the vulnerability-adjustable event types discussed above, the frequency-adjustable event types can comprise those raw event types for which a high historical event frequency between a given pair of machines is seen as a reliable indicator of non-maliciousness for the network or computer being monitored by the fusion engine 22. Alternatively, also similar to the vulnerability adjustable event types discussed above, in another exemplary embodiment (not shown) the context database 638 could instead comprise a list that identifies those raw event types for which a high historical event frequency between a given pair of machines for a network or computer being monitored by the fusion engine 22 is not seen as a reliable indicator of non-maliciousness, and historical event frequency can then be considered a trustworthy indicator of non-maliciousness for all other event types. In such a scenario, the list would identify those raw events where it is undesirable to adjust the priority status thereof based on historical event frequency.
- 25 Alternatively, in yet another exemplary embodiment (not shown), the context database 638 could also comprise both types of lists where one list would identify those raw event types for which frequency-based priority adjustment is allowed and the other would identify those raw event types for which frequency-based priority adjustment is not allowed. In this case a conflict resolution rule must also be established, so that if a particular event type appears in both lists, it is well-
- 30

WG 01041265

PCT/US04/12799

defined which entry will take precedence. Those skilled in the art will appreciate that other configurations of lists are not beyond the scope of the present invention.

Following step 1135, in decision step 1145, it is determined whether a match exists with the stored frequency-adjustable event types. If the inquiry to decision step 1145 is negative, then the "no" branch is followed to step 1210 of Figure 13. If the inquiry to decision step 1145 is positive, then the "yes" branch is followed to decision step 1150.

In decision step 1150, it is determined whether historical frequency information exists for the current raw event being evaluated. This determination is made based on the historical frequency value 506 of the raw event previously established by step 1015 of procedure 730 described in Figure 10. In other words, some raw events may be of a type, source, and destination that was not seen in the historical data analyzed to produce the historical frequency information. If the inquiry to decision step 1150 is negative, then the "no" branch is followed to step 1210 of Figure 13. If the inquiry to decision step 1150 is positive, then the "yes" branch is followed to decision step 1155.

In decision step 1155, it is determined whether the historical frequency for the current raw event being evaluated is greater than a frequent event threshold. In other words, in this decision step, it is determined whether a raw event is of a type that occurs frequently enough between a specific source and destination that it can be considered to be likely to be a non-malicious event. The frequent event threshold may be a value that corresponds to an average number of events per unit time, such as per day. However, other mathematically related forms of this value, such as the average time between events, could also be used and are not beyond the scope of the present invention. If the current raw event being processed has an historical event frequency that is greater than the threshold, then it is considered to be a frequent event and likely to be non-malicious.

If it is determined to be a frequent raw event, then its priority status can be lowered. However, if the current raw event being processed has been seen less frequently on the network, then it is not considered to be a frequent raw event and

WO 01/84285

PCT/US00/03799

an adjustment to its priority status based on historical event frequency is considered undesirable. Therefore, if the inquiry to decision step 1155 is negative (meaning that events like the current raw event being processed have not been seen frequently on the network monitored by the fusion engine 22), then the "no" branch is followed to step 1210 of Figure 12. If the inquiry to decision step 1155 is positive (meaning that events like the current raw event being processed have been seen frequently on the network monitored by the fusion engine 22), then the "yes" branch is followed to decision step 1160.

In decision step 1160, it is determined whether the raw event being processed is at its lowest priority status. If the inquiry to decision step 1160 is positive, then the "yes" branch is followed to step 1210, Figure 12. If the inquiry to decision step 1160 is negative, then the "no" branch is followed to step 1165, in which the priority of the current raw event is reduced and the reason for changing the status of the priority of the current raw event is recorded. The reason is typically recorded as being that the raw event being evaluated occurs frequently.

The process then continues to Figure 12. Figure 12 illustrates a second portion of the computer-implemented process for routine 735 of Figure 7, in which the CoBRA processor 625 adjusts the priority status or leaves an original priority status intact based upon the CoBRA assigned context parameters or detector-assigned type parameters of the raw event.

In step 1210, the raw event is compared to non-adjustable event types stored in a list of the context database 630. Similar to the vulnerability adjustable event types and frequency-adjustable event types discussed above, the non-adjustable event types are raw event types that may be defined by a network security administrator that are deemed to present low risk to the network or computer being monitored by the fusion engine 22 if they occur internally (that is, if both the source Internet protocol address and destination Internet protocol address in the raw event are located in networks defined in the context database 630 as belonging to the internal zone). However, in an alternative embodiment (not shown), the context database 630 may instead comprise a list that identifies

WO 01/4285

PCT/US88/13799

raw event types that cannot be deemed to present low risk to the network or computer being monitored by the fusion engine 22 based solely on the zone(s) in which the source and destination are located.

- In each an embodiment, event types other than those listed are deemed to present low risk to the monitored computer or network if they occur internally. In a further alternative embodiment (not shown), the context database 630 may also recognize both types of list: one list identifying raw event types that cannot be deemed to present low risk based solely on the source and destination zones, where the priority status thereof should not be adjusted, and a second list of raw events that are deemed to present low risk when seen internally, and where priority status should be adjusted such that these events will have a lower priority status. In this case a conflict resolution rule must also be established, so that if a particular event type appears in both lists, it is well-defined which entry will take precedence.
- 15 In decision step 1215, it is determined whether a match exists with the stored non-adjustable event types in the context database 630. If the inquiry to decision step 1215 is negative, then the "no" branch is followed back to routine 740 of Figure 7. If the inquiry to decision step 1215 is positive, then the "yes" branch is followed to decision step 1220.
- 20 In decision step 1220, it is determined whether the source zone and destination zone of the current raw event being processed are both internal relative to the network or computer being monitored by the fusion engine 22. This determination is made by examining the values of the source zone parameter 508 and destination zone parameter 510 of the raw event assigned by steps 1020 and 1025, respectively, of routine 730 shown in Figure 10. For many event types, raw events classified as being internal are less of a threat to a network of computers being monitored compared to an event that may be external to a network or computer being monitored by the fusion engine 22.
- 25 Therefore, for internal events, it may be desirable to lower the priority status of such raw events. Conversely for raw events for which either the source

WO 01/81285

PCT/00/013779

or destination Internet protocol address is either in the DMZ zone or not in any defined zone (and therefore considered external), it may be desirable to keep the priority status of a raw event that was assigned to it by the detectors in the intrusion detection system. If the inquiry to decision step 1220 is negative, then the "no" branch is followed back to routine 740 of Figure 7. If the inquiry to decision step 1220 is positive, then the "yes" branch is followed to decision step 1225.

In decision step 1225, it is determined whether the current raw event is at its lowest priority status. If the inquiry to decision step 1225 is positive, then the "yes" branch is followed back to routine 740, Figure 7. If the inquiry to decision step 1225 is negative, then the "no" branch is followed to step 1230, in which the priority status of the current raw event is reduced and the reasons for change in the priority status of the raw event is recorded. Typically, the reason in step 1230 will indicate that the priority status of the current raw event was lowered because it comprises an internal attack. The process then returns to routine 740 of Figure 7.

The present invention is also not limited to the technique of reducing priority status values. In other words, the present invention can also comprise a scale where values are increased in order to reflect either reduced priority or increased priority. Those skilled in the art will appreciate that any number of risk adjustment schemes can be utilized and not depart from the scope and spirit of the present invention.

Referring now to Figure 13, this Figure illustrates the computer-implemented process for routine 740 of Figure 7 in which raw events are associated with predetermined correlation rules based upon the event type parameter 555. In this routine, the classifier 615 may identify one or more correlation rules 620 that should process each given raw event. Step 1310 is the first step of routine 740, in which all lists containing the raw events are updated to reflect any CoBRA processing changes. In other words, all objects in the exemplary object-oriented architecture containing the raw events that were

WO 01/04285

PCT/US98/13799

adjusted by the CORR processor 625 are updated to reflect any adjustments in priority status.

Next, in step 1315, the raw events are forwarded to the correlation rules that apply to the raw event type parameter 555. More specifically, in step 1315, the definition of each correlation rule 620 includes a list of the raw event categories that are of interest to it. The raw event types included in each raw event category are defined in the raw event classification database 635. Therefore, the list of raw event types of interest to a correlation rule 620 is the union of the category-specific lists of raw event types for the categories of interest to the rule, where each category-specific list of raw event types is defined by the raw event classification database 635. The category-specific lists of raw event types are stored in the classifier 645, which is initialized based on the contents of the raw event classification database 635.

When the controller 655 loads a correlation rule 620 during system initialization, it associates the rule with all the event types included in the event categories of interest to the rule (determined as described in the previous paragraph) by adding the rule to a list of interested rules maintained within each such event type. Thus, after initialization, each event type includes a list of all of the correlation rules 620 that are interested in events of its type. As each raw event is received, the event reader 660 determines which correlation rules 620 should process it by retrieving the raw event's event type and then retrieving the event type's list of interested rules. Having determined the set of correlation rules 620 that should process the raw event, the process then returns to step 745 of Figure 7.

Referring now to Figure 14, this Figure illustrates the computer-implemented process for routine 770 of Figure 7, which determines if a currently immature correlation event to which the current processed raw event has been added meets or satisfies the maturity criteria of a corresponding rule 620. The process described here is for an exemplary event type, Attack From Attacked Host (AFAH), rather than being generic. However, given this processing description and the descriptions of exemplary correlation event types presented

WO 01/62805

PC20001/03799

urther, it should be apparent to those skilled in the art how similar processing could be used to recognize the occurrence of each of the described exemplary event types. As noted above, such rule 628 may be implemented as a rule object in an object-oriented computer architecture. As should be clear to those skilled in the art from the previously described processing of step 1315 of Figure 13, a single raw event may be processed by multiple correlation rule objects.

Though not depicted in Figure 7 or Figure 14, the processing of steps 745 through 780 of Figure 7 (including the routine 778 processing described in Figure 14) can be performed twice for the current processed raw event in the case of the exemplary AFAH correlation event. In one exemplary embodiment, the raw event is processed once to consider it as an inbound attack only if the raw event is an integrity attack, and is always processed to consider it as an outbound attack. When the raw event is considered as an inbound attack by steps 745 through 780, step 745 uses the raw event's destination Internet protocol address as the lookup key when attempting to retrieve a corresponding AFAH correlation event from the correlation event cache 645 (as described earlier in the discussion of step 745 processing).

When the raw event is considered as an outbound attack by steps 745 through 780, step 745 uses the raw event's source Internet protocol address as the lookup key when attempting to retrieve a corresponding AFAH correlation event from the correlation event cache 645 (as also described earlier in the discussion of step 745 processing). This "double processing" of the raw event is a unique aspect of the exemplary AFAH correlation event relative to other correlation events that can be processed by the fusion engine 22. For all of the other exemplary correlation event types described earlier, the processing of steps 745 through 780 is performed once, as should be apparent to those skilled in the art based on the description of the exemplary correlation event types.

Referring again to Figure 14, step 1410 is the first step of routine 778, in which the event type object of the identifier 615 for the current raw event being processed is added to the raw event tracking index 646. Also, the correlation

WO 01/94285

PCT/US01/13799

event object corresponding to the current raw event object being processed is either added to the memory management list 640 (if it is a new correlation event that was just created in step 750 of Figure 7), or is moved to a new position in the memory management list if the timestamp of the current raw event has the most recent timestamp of all the raw events associated with the current correlation event.

In addition, the current correlation event object is added to the raw event tracking index 645 in association with the raw event. The event type object and correlation event object are added to the raw event tracking index 645 so that they can later be informed by the memory management processing if the raw event is erased from memory, so they can erase their own references to the raw event. The current correlation event is also added to the memory management list 640 so that when memory resources run low, the oldest events (some of which may be immature correlation events) can be deleted from the fusion engine 22.

In decision step 1415, it is determined whether the raw event is being considered as an inbound attack. This step distinguishes whether the current AFAH correlation event includes the current raw event as an inbound or outbound attack. If the inquiry to decision step 1415 is negative, then the raw event is being considered as an outbound attack and the "no" branch is followed to step 1425. If the inquiry to decision step 1415 is positive, then the raw event being considered is an inbound attack and the "yes" branch is followed to step 1420.

In decision step 1420, it is determined whether the raw event being considered as an inbound attack is an integrity attack and occurs earlier than at least one event in the outbound attacks list of the current correlation event. The raw event being processed is known to be an integrity attack since it was added to the inbound attack list of the correlation event during the processing of step 1415. As indicated in the description of the Attack From Attacked Host event included in the *sa/rev* list of exemplary correlation event types, the AFAH event can be generated when an integrity attack is seen against a host followed by a

Confidentiality, Integrity, or Availability attack originating from that host. As indicated in the discussion of Figure 13, when the controller 655 loads a

WO 01/84283

PCT/US96/13799

correlation rule 620 during system initialization, it associates the rule with all of the event types included in the event categories of interest to the rule.

In the case of the AFARI rule, the event categories of interest are Confidentiality, Integrity, and Availability attacks. The AFARI rule is therefore associated with all event types defined by the raw event classification database 635 as belonging to one of these three categories. Therefore, any raw event whose event type belongs to one of these three categories can be forwarded to routine 770 for processing. Since the definition of the AFARI event requires inbound attacks to be Integrity attacks, and some of the raw events forwarded to routine 770 can instead be Confidentiality or Availability attacks, the present decision step 1420 must verify that the raw event being considered as an inbound attack is an Integrity attack.

A further consideration results from the fact that the fusion engine 22 can receive raw events generated by multiple detectors, leading to the possibility that raw events can be received in non-chronological order (that is, a raw event with a later timestamp can be received before a raw event with an earlier timestamp). For this reason, routine 770 cannot assume that raw events will be received in chronological order, and the present decision step 1420 therefore determines whether the current raw event occurs earlier than at least one event in the outbound attacks list of the current correlation event. If the inquiry to decision step 1420 is negative, then the current correlation event is deemed not mature and the "no" branch is followed back to routine 785 of Figure 7. If the inquiry to decision step 1420 is positive, then the current correlation event is deemed mature and the "yes" branch is followed to step 1427.

In decision step 1425, it is determined whether the raw event being considered as an inbound attack occurs later than at least one event in the inbound attacks list of the current correlation event. Unlike decision step 1420, it is not necessary to determine whether the current raw event belongs to a particular category because (as described in the discussion of step 1420) every raw event forwarded to routine 770 will be either a Confidentiality, Integrity, or Availability attack and will therefore meet the criteria for inclusion as an inbound attack in

WO 01/84285

PCT/JP98/01799

any AFAH event. If the inquiry to decision step 1425 is negative, then the current correlation event is deemed not mature and the "no" branch is followed back to routine 785 of Figure 7. If the inquiry to decision step 1425 is positive, then the current correlation event is deemed mature and the "yes" branch is followed to step 1427.

In step 1427, any outboard attacks in the current correlation event that occur earlier than the earliest inbound attack of the correlation event are removed from the list of outboard attacks. This is done because the definition of the AFAH correlation event requires that each outboard attack included in a mature AFAH correlation event must be preceded by at least one inbound attack.

In step 1430, the correlation event is removed from the memory management list 640 so that the correlation event will not be subject to being erased by the memory management mechanisms. In this way, the correlation event that is removed will not be deleted from the fusion engine 22 since the correlation event is now deemed to be mature.

In step 1435, the update time of the correlation event can be set to the most recent raw event's time stamp if the event source being read by the event reader 600 is either the event database 26 or the event log file 610, in which case the fusion engine 22 is operating in a batch mode. Alternatively, the update time of the correlation event can be set to the current time of the system on which the fusion engine 22 is executing if the event source being read by the event reader 600 is the event collector 24, in which case the fusion engine 22 is operating in a real-time mode.

In step 1440, the correlation event is added as the mature event correlation list 650. In step 1445, the correlation event containing the two or more raw events is indicated as being mature by setting an internal parameter of the correlation event. The process then returns to step 775 of Figure 7. In one exemplary embodiment (not shown), each correlation event may be assigned a priority status, similar to the priority status parameter 555 of raw events.

WG 018/285

PCT/US01/13799

Example Rule Processing for Raw Event II Illustrated in Figures 5D, 5E, 5F

The following is the processing that would be carried out by an Attack From Attacked Host correlation rule 620 for raw event II as illustrated in Figures 5D, 5E, and 5F. This discussion assumes that raw events I and II are both types of integrity attacks and therefore qualify as inbound attacks according to the definition of the AFAH event, that raw event I occurs before raw event II, and that raw event II occurs before raw event III.

Referring back to Figure 7, in step 745, when raw event II is being considered as an inbound attack, its destination internet protocol address (3.3.3.3) would be used as a lookup key to retrieve an AFAH correlation event from the correlation event cache 665. Assuming that in this case raw events are received in chronological order and therefore raw event III has not yet been processed by the fusion engine, there would be no AFAH correlation event indexed by the attacked internet protocol address 3.3.3.3, and therefore the "no" branch of decision step 745 would be taken and correlation event 513 would be created in step 750. In step 755, correlation event 513 would be stored in the correlation event cache 665. In step 760, raw event II would be associated with correlation event 513 by storing a reference to it in the inbound attacks list of correlation event 513. In step 765 it would be determined that correlation event 513 is not already mature, so the "no" branch would be followed to step 770.

Referring now to Figure 14, in decision step 1415 the "yes" branch would be followed since raw event II is being considered as an inbound attack. In decision step 1430, the "no" branch would be followed since there are no raw events in the outbound attacks list of the newly created correlation event 513. To summarize this processing, when considered as an inbound event, raw event II is added to a newly created but still immature correlation event 513.

Referring back to Figure 7, in step 745, when raw event II is being considered as an outbound attack, its source internet protocol address (2.2.2.2) would be used as a lookup key to retrieve an AFAH correlation event from the correlation event cache 665. Assuming that in this case raw events are received in

WO 01/64285

PCT/JP98/13799

chronological order and therefore raw event I has already been processed by the fusion engine. AFAT correlation event 511 would already be present in the correlation event cache 665 indexed by the attached internet protocol address 2.2.2.2, and therefore the "yes" branch of decision step 745 would be taken to step 768. In step 768, raw event II would be associated with correlation event 511 by storing a reference to it in the outbound attacks list of correlation event 511. In step 765 it would be determined that correlation event 511 is not already mature, so the "no" branch would be followed to step 778.

Referring now to Figure 14, in decision step 1415 the "no" branch would be followed since raw event II is being considered as an outbound attack. In decision step 1425, the "yes" branch would be followed since the inbound attacks list of correlation event 511 already contains raw event I, and the timestamp of raw event II is later than that of raw event I. At this point, correlation event 511 has been determined to be mature and steps 1427 through 1445 would be followed to process the newly-mature correlation event 511. To summarize this processing, when considered as an outbound attack, raw event II is added to existing correlation event 511 which becomes mature as a result.

To perform decision step 1425 in the above-described exemplary rule processing, the respective time stamps of the first-generated raw event I and the second-generated raw event II are compared. However, it is noted that since the raw events could originate from different detectors, there could be some variance in the time stamps provided for each raw event. That is, while the second-generated raw event II may occur after the first-generated raw event I, because of possible variances in the internal clocks of the detectors generating the raw events, it is foreseeable that the first-generated raw event I may have a later time stamp than the second-generated raw event II.

In other words, the internal clocks between respective detectors in neighboring intrusion detection systems may not be synchronized. In order to compensate for such a scenario, a tri-state comparison could be performed. That is, the fusion engine 22 and more specifically, the rules 618 may allow for the possibility that there may be some synchronization efforts so a determination can

WO 01/04285

PCT/US98/13799

be made if a first raw event came before another raw event. More specifically, when comparing the timestamps of two raw events generated by different detectors to determine whether one of the events precedes (or follows) the other, the result of the comparison can be yes, no, or maybe. The "maybe" result occurs when the timestamps of the two events are sufficiently close that uncertainty regarding the synchronization offsets of the two detectors makes it impossible to determine which event occurred first.

The fusion engine 22 could be configured to treat the "maybe" result either as a "yes" (in one configuration) or as a "no" (in an alternative configuration). In a preferred embodiment, the fusion engine 22 treats a "maybe" as a "yes" to maximize the chance that correlation event maturity criteria will be met (so that mature correlation events will be generated whenever it appears possible that their criteria might be met). When the fusion engine 22 compares the timestamps of two events generated by the same detector, then it can ignore any synchronization effects and perform a simple binary comparison between the timestamps of the two events.

Exemplary Computer-Implemented Process to Determine if Mature Correlation Events Timed Out

Referring now to Figure 15, this Figure illustrates the computer-implemented process for routine 785 which determines whether any mature correlation events have stopped occurring. Step 1530 is the first step of routine 785 in which the current processing time is compared with the update times of the correlation events stored in the mature event list 650 (the update times of correlation events are set as described in step 1435 of figure 14). For the purpose of this comparison, the definition of the current processing time depends on the mode in which the fusion engine 22 is operating. If it is operating in a batch mode (that is, the event source from which events are being read is either the event database 26 or the event log file 410), then the current processing time is the timestamp of the last event that was read from the event source. Alternatively, if the fusion engine 22 is operating in real-time mode (that is, the event source from

W/O 01/84285

PCT/US01/17799

which events are being read in the event collector 24), then the current processing time is the current time of the system on which the fusion engine 22 is running.

In decision step 1515, it is determined whether the difference between the current processing time and the update time of each correlation event exceeds a predetermined threshold. In other words, it is determined whether the mature correlation events contained within the mature event list 650 have become old or stale in that no computer activity or new events have occurred for these correlation events over some period of time. If the inquiry to decision step 1515 is positive, then the "yes" branch is followed back to step 790 of Figure 7. If the inquiry to decision step 1515 is negative, then the "no" branch is followed back to step 795 of Figure 7.

It should be understood that the foregoing relates only to illustrative embodiments of the present invention, and that numerous changes may be made therein without departing from the spirit and scope of the invention as defined by the following claims.

WO 01/04185

PCT/US01/3799

CLAIMS

What is claimed is:

1. A method for managing security information comprising the steps of:
5 receiving raw events from one or more data sources;
classifying the raw events;
storing the raw events;
assigning a ranking to each raw event;
identifying relationships between two or more raw events;
10 is responsive to identifying any relationships between two or more raw events, generating a mature correlation event message; and
displaying one or more mature correlation event messages on a console that describe relationships between raw events.
- 15 2. The method of claim 1, wherein each raw event comprises suspicious computer activity detected by one of an automated system and human observation.
3. The method of claim 1, wherein the step of receiving raw events from one or more data sources further comprises the step of receiving real-time raw events
20 from one of intrusion detection system, a detector within an intrusion detection system, and a firewall.
4. The method of claim 1, wherein the step of receiving raw events from one or more data sources further comprises the step of receiving raw events from one of
25 a file and database.
5. The method of claim 1, wherein the step of classifying the raw events further comprises the steps of:
30 identifying an event type parameter for each raw event;
comparing the event type parameter with an event type category of a list;
and

WO 01/4285

PCT/US97/03799

assigning each raw event to a corresponding event type category in the list.

6. The method of claim 1, wherein the step of assigning a ranking to each raw event further comprises the steps of:
- 5 computing parameters of each raw event with information in a database; and
- assigning additional parameters to each raw event relating to the environment of the raw event.
7. The method of claim 6, wherein the additional parameters comprise one of a priority status, a vulnerability status, a historical frequency value, a source zone value, a destination zone value, a detector zone value, and a text string.
8. The method of claim 1, wherein the step of assigning a ranking to each raw event further comprises the steps of:
- 15 identifying a priority status parameter of a raw event;
- comparing each raw event to information contained in a context database;
- changing the priority status parameter of a respective raw event if a match occurs in response to the comparison step; and
- 20 leaving the priority status in tact if a match does not occur in response to the comparison step.
9. The method of claim 1, wherein the step of identifying relationships between two or more raw events further comprises the steps of:
- 25 associating each raw event with a rule which corresponds with a type parameter of a raw event; and
- applying one or more rules to groups of raw events having the same type parameter; and
- determining if a computer attack or security breach has occurred based
- 30 upon successful application of a rule.

WO 01/4285

PCT/US00/17799

10. The method of claim 1, wherein the step of storing raw events further comprises the step of storing each raw event in a high speed memory device comprising random access memory (RAM).
- 5 11. The method of claim 1, further comprising the step of determining the intent of a computer attack based upon the type of malware execution event generated.
12. The method of claim 1, further comprising the steps of:
creating a memory management list;
10 identifying a time stamp for each raw event; and
adding each raw event to the memory management list.
13. The method of claim 1, further comprising the step of creating a raw event tracking index that identifies one or more software components that are
15 monitoring one or more raw events.

WO 01/42485

PCT/US01/17795

14. A method for determining relationships between two or more computer events, comprising the steps of:
receiving a plurality of raw events having a first set of parameters;
creating raw event storage areas based upon information received from a
5 raw event classification database;
storing each event in an event storage area based upon an event type parameter;
comparing each raw event to data contained in a context database;
adjusting a priority parameter or leaving the priority parameter in tact for
10 each raw event in response to the comparison to the context database;
associate each raw event with a correlation event;
applying one or more rules to each event based upon the correlation event association; and
generating a mature correlation event message in response to a successful
15 application of a rule.
15. The method of claim 14, wherein each raw event comprises suspicious computer activity detected by one of an automated system and human observation.
- 20
16. The method of claim 14, wherein the context database comprises any one of vulnerability values, computer event frequency values, and source and destination zone values.
- 25
17. The method of claim 14, wherein the raw event classification database comprises tables that include information that categorize raw events based on any one of the following: how a raw event may impact one or more target computers, how many target computers that may be affected by a raw event, and how respective raw events gain access to one or more target computers.

WO 01/84285

PCT/JP98/03799

18. A security management system comprising:
a plurality of data sources;
an event collector linked to the plurality of data sources;
a fusion engine linked to the event collector, said fusion engine identifying
relationships between two or more raw events generated by the data sources; and
a console linked to the event collector for displaying any output generated
by the fusion engine.
19. The security management system of claim 18, further comprising a detector,
the detector running in a kernel mode of a computer and the fusion engine running
in a user mode of the computer.
20. The security management system of claim 18, further comprising a detector
chip, and the fusion engine comprising software running on a computer.
21. The security management system of claim 18, further comprising a detector
board, and the fusion engine comprising software running on a computer.

WO 01/84286

PCT/JP01/03799

22. A fusion engine comprising:
a controller;
an event reader for receiving raw events;
a classifier linked to the event reader for classifying the received raw
5 events;
a raw event classification database linked to the classifier;
a context based risk-adjustment processor linked to the classifier, for
adjusting priorities of raw events;
a context database linked to the context based risk-adjustment processor;
10 and
a rule database, for determining if relationships exist between two or more
events.
23. The fusion engine of claim 22, further comprising an event reporter, a mature
15 events list, a memory management list, and a raw event tracking index.
24. The fusion engine of claim 22, wherein the context database comprises any
one of vulnerability values, computer event frequency values, and source and
destination zone values.
- 20 25. The fusion engine of claim 22, wherein the raw event classification database
comprises tables that include information that categorizes raw events based on any
one of the following: how a raw event may impact one or more target computers,
how many target computers that may be affected by a raw event, and how
25 respective raw events gain access to one or more target computers.

WU 41/84285

PCT/JP04/03799

2/12

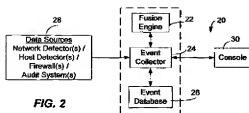


FIG. 2

FIG. 3

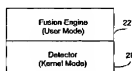
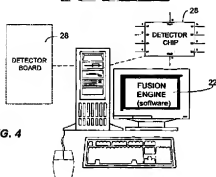


FIG. 4



3/12

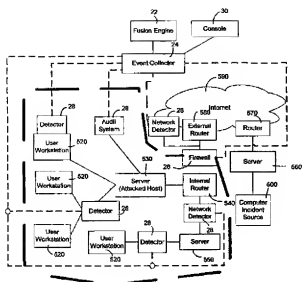


FIG. 5A

W O 2004/01287

PCT/501/3799

4/12

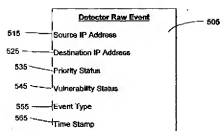


FIG. 5B

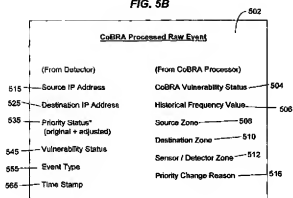


FIG. 5C

5/12

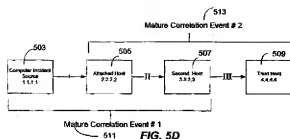


FIG. 5D

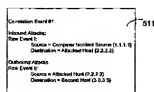


FIG. 5E

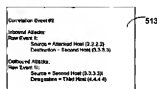
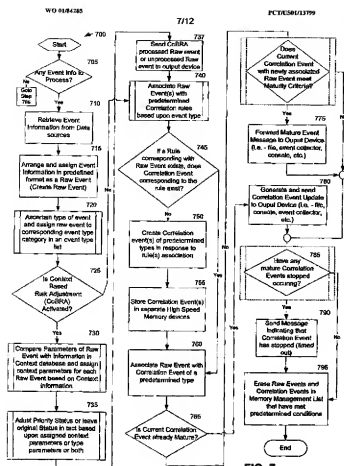


FIG. 5F



WO 01/94265

PCT/00/012799

8/12

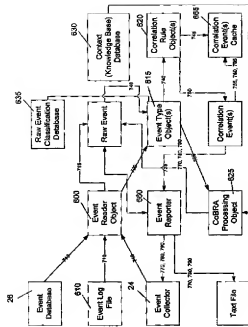


FIG. 8

WO 01/4285

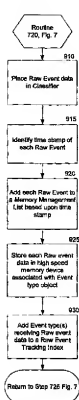
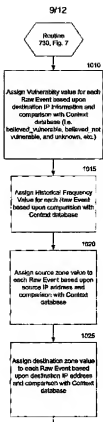


FIG. 9

9/12



PCT/US99/03799

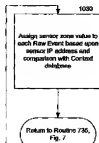


FIG. 10

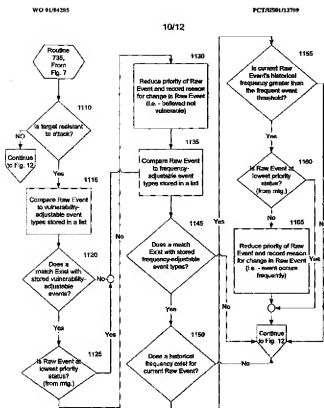
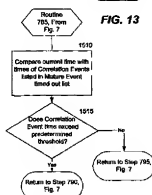
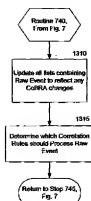
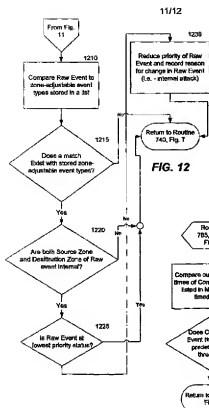


FIG. 11

WO 01/04285

PLT/EDU/AL779



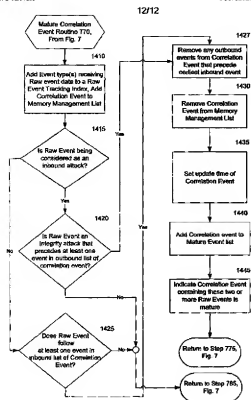


FIG. 14

WO 01/84285 A3



(98) Date of publication of the international search report: For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT				Inventor Application No.	
Information on prior art documents					
Patent & Non-patent literature		Publication date	Patent family member(s)	Publication date	
EP 0981995	A	15-03-2000	EP 0985995 A1	15-03-2000	

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CN,GA,GN,GW,ML,MR,NE,SN,TD,TC),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DW,DZ,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZW

(特許庁注：以下のものは登録商標)

U N I X

(72)発明者 ハンマー、 ジョン エム。

アメリカ合衆国 ジョージア州 3 0 0 9 2 ノークロス ウィルマー ドライブ 5 5 8 4

(72)発明者 ウィリアムズ、 ブライアン ダグラス

アメリカ合衆国 ジョージア州 3 0 0 4 3 ローレンスヴィル ソーンツリー パス 4 3 0

(72)発明者 ブラス、 フィリップ チャールズ

アメリカ合衆国 ジョージア州 3 0 0 7 5 ロスウェル パイン グローブ ポイント ドライブ 1 1 4 0

(72)発明者 ヤング、 ジョージ スイー。

アメリカ合衆国 ジョージア州 3 0 0 9 2 ノークロス コモンズ ゲート ベンド 3 3 5 5

(72)発明者 メザック、 デレク ジョン

アメリカ合衆国 ジョージア州 3 0 0 6 6 マリエッタ ブラックウェル ラン 3 6 1 5

Fターム(参考) 5B042 GA12 GC08 MA08 MC09 MC40

5B085 AA08 AC13 AC16 BG02

5B089 KA17 KB10 KB13 KC39

【要約の続き】

。

【選択図】図6

【公報種別】特許法第17条の2の規定による補正の掲載

【部門区分】第6部門第3区分

【発行日】平成20年2月28日(2008.2.28)

【公表番号】特表2004-537075(P2004-537075A)

【公表日】平成16年12月9日(2004.12.9)

【年通号数】公開・登録公報2004-048

【出願番号】特願2001-580642(P2001-580642)

【国際特許分類】

G 0 6 F 11/34 (2006.01)

G 0 6 F 11/32 (2006.01)

G 0 6 F 21/20 (2006.01)

G 0 6 F 13/00 (2006.01)

【F I】

G 0 6 F 11/34 L

G 0 6 F 11/32 L

G 0 6 F 15/00 3 3 0 A

G 0 6 F 13/00 3 5 1 Z

【手続補正書】

【提出日】平成20年1月10日(2008.1.10)

【手続補正1】

【補正対象書類名】明細書

【補正対象項目名】特許請求の範囲

【補正方法】変更

【補正の内容】

【特許請求の範囲】

【請求項1】

1 以上のデータ・ソースから生イベントを受け取るステップと、
前記生イベントを分類するステップと、
前記生イベントを記憶するステップと、
各生イベントに順位を付けるステップと、
2 以上の生イベントの間に関係を見つけて出すステップと、
2 以上の生イベントの間に何らかの関係を見つけて出すことに対応して、成熟相関イベント・メッセージを生成するステップと、
生イベントの間の関係を記述する1以上の成熟相関イベント・メッセージをコンソールに表示するステップと、
を含む、セキュリティ情報を管理するための方法。

【請求項2】

前記1以上のデータ・ソースから生イベントを受け取るステップが、侵入探知システム、侵入探知システム内のディテクタ、およびファイヤ・ウォールのうちの1つからリアルタイムの生イベントを受け取るステップをさらに含む請求項1記載の方法。

【請求項3】

前記1以上のデータ・ソースから生イベントを受け取るステップがファイルおよびデータベースの1つから生イベントを受け取るステップをさらに含む請求項1記載の方法。

【請求項4】

前記生イベントを分類するステップが、
各生イベントのためのイベント・タイプ・パラメータを見つけて出すステップと、
前記イベント・タイプ・パラメータをリストのイベント・タイプ・カテゴリと比較するステップと、

各生イベントを前記リスト中の対応するイベント・タイプのカテゴリに配属するステップと、

をさらに含む請求項 1 記載の方法。

【請求項 5】

前記各生イベントに順位をつけるステップが、
各生イベントのパラメータをデータベース中の情報と比較するステップと、
各生イベントに前記生イベントの環境に関係する追加のパラメータを付け加えるステップと、

をさらに含む請求項 1 記載の方法。

【請求項 6】

前記追加のパラメータが優先順位、脆弱度、ヒストリカル頻度値、送信元ゾーン値、宛先ゾーン値、およびテキスト列のどれか 1 つを含む請求項 5 記載の方法。

【請求項 7】

前記各生イベントに順位をつけるステップが、
生イベントの優先順位パラメータを見つけ出すステップと、
各生イベントをコンテキスト・データベースに含まれている情報と比較するステップと、

前記比較ステップに応じて一致が発生したときは各生イベントの優先順位パラメータを変更するステップと、

前記比較ステップに応じて一致が発生しなかったときは優先順位をそのままにするステップと、

をさらに含む請求項 1 記載の方法。

【請求項 8】

前記 2 以上の生イベントの間の関係を見つけ出すステップが、
各生イベントを生イベントのタイプ・パラメータに対応する 1 以上のルール と関係付けるステップと、

各ルールを生イベントの関係付けられたグループに適用するステップと、 成功したルールの適用に基づいてコンピュータに対する攻撃またはセキュリティの侵害が起きたかどうか判断するステップと、

をさらに含む請求項 1 記載の方法。

【請求項 9】

前記生イベントを記憶するステップが、各生イベントをランダム・アクセス・メモリ (RAM) を含む高速メモリ・デバイスに記憶させるステップをさらに含む請求項 1 記載の方法。

【請求項 10】

前記生成された前記成熟関連イベントのタイプに基づいてコンピュータに対する攻撃の意図を判断するステップをさらに含む請求項 1 記載の方法。

【請求項 11】

メモリ管理リストを作成するステップと、
各生イベントのタイムスタンプを見つけるステップと、
各生イベントを前記メモリ管理リストに付け加えるステップと、
をさらに含む請求項 1 記載の方法。

【請求項 12】

1 以上の生イベントを監視している 1 以上のソフトウェア・コンポーネントを見つけ出すための生イベント追跡インデックスを作成するステップをさらに含む請求項 1 記載の方法。

【請求項 13】

第 1 の組のパラメータを持つ複数の生イベントを受け取るステップと、
生イベント分類データベースから受け取った情報に基づいて生イベント記憶領域を設けるステップと、

各イベントをイベント・タイプ・パラメータに基づいてイベント記憶領域に記憶するステップと、

各生イベントをコンテキスト・データベース中に含まれるデータと比較するステップと、

コンテキスト・データベースとの前記比較に応じて各生イベントのための優先度パラメータを調整するかまたはそのままにするステップと、

各生イベントを1以上の関連イベントに関係付けるステップと、

前記関連イベントの連合に基づいて各イベントに1以上のルールを適用するステップと、

各成功したルールの適用に応じて成熟関連イベント・メッセージを生成するステップと、

を含む、2以上のコンピュータ・イベントの間の関係を判断するための方法。

【請求項14】

前記コンテキスト・データベースが、脆弱度値、コンピュータ・イベントの頻度値、送信元および宛先ゾーン値、およびディテクタ・ゾーン値のどれか1つを含む請求項13記載の方法。

【請求項15】

前記生イベント分類データベースが、次の情報、すなわち、生イベントによって示される活動が1以上の標的コンピュータにどのようにして影響を与える可能性があるか、生イベントによって示される活動によって影響を受ける可能性があるコンピュータの数はどのくらいか、および各生イベントによって示される活動はどのようにして1以上の標的コンピュータにアクセスするか、のどれかに基づいて生イベントをカテゴリに分ける情報を含む請求項13記載の方法。

【請求項16】

複数のデータ・ソースと、

複数のデータ・ソースにリンクされたイベント・コレクタと、

前記イベント・コレクタにリンクされたフュージョン・エンジンであって、前記データ・ソースによって生成された2以上の生イベントの間の関係を見つけたフュージョン・エンジンと、

前記イベント・コレクタにリンクされた、前記フュージョン・エンジンによって生成された出力を表示するためのコンソールと、

を含むセキュリティ管理システム。

【請求項17】

コンピュータのカーネル・モードで動作するディテクタと前記コンピュータのユーザ・モードで動作する前記フュージョン・エンジンをさらに含む請求項16記載のセキュリティ管理システム。

【請求項18】

ディテクタのチップ、およびコンピュータ上で動作するソフトウェアを含む前記フュージョン・エンジンをさらに含む請求項16記載のセキュリティ管理システム。

【請求項19】

ディテクタ基板、およびコンピュータ上で動作するソフトウェアを含む前記フュージョン・エンジンをさらに含む請求項16記載のセキュリティ管理システム。